The Design and Safety Analysis of an Accessible Low-Cost Differential Drive Robot Under LIDAR-Based Obstacle Avoidance

Approved by

Supervising Committee:

Raul G. Longoria, Supervisor

Junmin Wang

The Design and Safety Analysis of an Accessible Low-Cost Differential Drive Robot Under LIDAR-Based Obstacle Avoidance

by

Natali V Salas-España

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

The University of Texas at Austin

May 2021

Dedicated to my mother, Maria, and to my brother, Angel.

Acknowledgments

First and foremost, I would like to thank my mother and my brother for providing me with their unconditional support throughout my studies. As a first-generation, Mexican-American student, a formal education at the highest level is a privilege afforded to me only through the struggles and sacrifices of my family. Through them, I am able to pursue my passion for engineering and robotics. For this, I am grateful.

I would like to express my gratitude to my research advisor and mentor, Dr. Raul Longoria. He has consistently provided me with support and guidance throughout the various stages of my research. Additionally, his encouraging nature has allowed me to explore a variety of research and industry interests that have been fundamental in shaping my career as an engineer.

To my lab mates, Suraj Pawar, Ethan Rapp, and Parthasarathi Ainampudi, thank you for your guidance and willingness to help, no matter how simple my questions. And to my colleagues, Vibhav Gaur and Vasko Lalkov, thank you for your company and discourse during our long study nights.

The Design and Safety Analysis of an Accessible Low-Cost Differential Drive Robot Under LIDAR-Based Obstacle Avoidance

Natali V Salas-España, M.S.E
The University of Texas at Austin, 2021

Supervisor: Raul G. Longoria

A low-cost mobile robot, OpenDani, is re-designed and built to serve as a research platform capable of autonomous tasks. A self-contained, open-source design is supported by the single-board computer mounted atop the robot. Low-cost sensors are integrated into the robot communication network to achieve online obstacle avoidance (OA). The mobile system is compared to its previous, non-autonomous design to highlight an elevated level of autonomy. An experimental analysis is performed to define a relationship between components of the OpenDani design and the safe operation of the vehicle under an obstacle avoidance task.

Table of Contents

Acknowledgments **v**

Abstract **vi**

List of Tables **x**

List of Figures **xii**

Chapter 1 Introduction **1**

1.1 Objectives 1

1.2 Motivation and Justification 1

Chapter 2 OpenDani Model Overview **4**

2.1 Introduction 4

2.2 Differential Drive System Model 5

2.3 OpenDani Wheel Velocity Mappings 7

2.4 Configuration Space and Non-Holonomic Constraints 9

Chapter 3 Autonomous Robot Design **11**

3.1 Introduction 11

3.2 Analyzing Autonomy 12

3.3 Review of Previous vDani System 13
3.3.1 Self-Localization 16
3.3.2 Perception 16
3.3.3 Planning 17
3.3.4 Trajectory Tracking 17
3.4 Improvements to the System 18
3.5 Cost Overview 19
3.6 Overview of the OpenDani System 20
3.6.1 Self-Localization 22
3.6.2 Perception 22
3.6.3 Planning 24
3.6.4 Trajectory Tracking 25

Chapter 4 Low-level Control **26**

4.1 Introduction 26
4.2 Finite-Difference Estimation for Wheel Speed 28
4.3 Kalman-Filter Theory 30
4.4 Kalman Filter Based Wheel Speed Estimation 32

Chapter 5 Obstacle Avoidance **36**

5.1 Introduction 36
5.2 Building a Point Cloud from LIDAR data 37
5.3 Sorting the Point Cloud to Identify Obstacles 39
5.4 Kinematically Feasible Paths 41
5.5 Greedily Choosing a Path 42
5.5.1 Free Path Length 43

5.5.2 Clearance 45
5.5.3 Distance to Goal 46
5.6 Path Execution via 1D Time Optimal Control 48

Chapter 6 Safety Experiments **52**
6.1 Introduction 52
6.2 Evaluating System Actuation Latency 54
6.2.1 Latency Compensation 57
6.2.2 Latency Compensation Drive Test Experiment 57
6.3 Safety Bumper Dimensions Based On Motion Uncertainty 59
6.3.1 Collision Experiments for Evaluating Safety Methods 64
6.4 Obstacle Avoidance Considering Motion Uncertainty 65
6.5 LIDAR Range Requirement From Robot Design Characteristics . . . 71
6.5.1 A Practical Application of the Minimum LIDAR Range Requirement 76
6.6 Verification of Minimum LIDAR Range Requirement 77
6.7 Obstacle Avoidance with Varying Environmental Uncertainty 81
6.7.1 On Perception Uncertainty 85

Chapter 7 Conclusion and Future Work **86**

Bibliography **88**

List of Tables

3.1 The vDani and OpenDani design mainly differ in their sensor suite and computational specifications. The OpenDani's Raspberry Pi contains a ROS installation that interfaces well with open-source software. 18

3.2 The vDani and OpenDani share several hardware components. These shared components make-up a base-cost for the two designs. 19

3.3 As the robot designs deviate from each other, so does their total design cost. The additional cost of each robot summarizes the cost in design differences. 20

6.1 An external, computer-vision based measurement system is utilized to compare against the odometry based measurements. The measured state is the distance traversed by the OpenDani vehicle during a 1 m test. 58

6.2 It is critical that wheel speed controller performance be examined for each unique set of robot operating parameters. The standard deviations of wheel speed controller errors are expected to vary with these operating parameters. 62

6.3 An obstacle is placed in the OpenDani's path for a series of maneuvers. Then, the two compensation methods for robot motion uncertainty are added and removed to provide various combinations. . . . 64
6.4 The OpenDani operating parameters correspond to the values utilized for LIDAR range analysis and the drive characteristics provided to the vehicle's motion algorithm during obstacle course experiments. . 78
6.5 A separate set of LIDAR ranges are examined for the straight driving and minimum turning radius stopping maneuver. 79
6.6 Isolated collision experiments for a variety of OpenDani LIDAR Ranges support the LIDAR range analysis that determines $Range_{min}$. 80

List of Figures

2.1 A reference frame and coordinate representation for a differential drive robot. Source:[1] 5
2.2 A rigid body with constrained movement to a plane. Source:[2] . . . 9

3.1 Autonomy levels for an AV are based on the amount of human driver intervention necessary to operate the vehicle. Level 5 autonomy requires no human intervention under all conditions to perform all driving functions. Source: [3] 13
3.2 The previous robot design, vDani, contains only a paper AprilTag atop it's body. Source: [4] 14
3.3 The system architecture of the vDani system relies on the communication between various ROS nodes. A camera routes images into an AprilTag detection node using the ROS communication network. Source: [4] 14
3.4 The OpenDani has several components mounted atop it's body. The components include a LIDAR sensor, a single-board computer, and a DC buck converter. A RedBoard micro-controller is not visible. . . . 21

3.5 A map of the OpenDani system architecture depicts the data transfer between software nodes for an obstacle avoidance functionality. . . . 21

4.1 A diagram shows the power transfer and data transfer between the low-level control components. An I2C communication protocol bridges the communication gap between the OpenDani Linux computer and the RedBoard micro-controller. 27

4.2 A simple step input test is performed on one of the OpenDani wheels. Upon reaching a steady-state condition, quantization noise is observed around the set point speed. 29

4.3 Quantization noise from the low-resolution wheel encoders introduces the need for signal processing methods in the OpenDani low-level control. The finite-difference speed estimate is replaced by a KF-based speed estimate to improve odometry. 34

4.4 A recursive discrete KF formulation is implemented directly on the RedBoard. The filtered speed estimate is used as feedback into the wheel speed controller. 35

5.1 The RPLIDAR A1 coordinate frame is useful for building a point cloud from angle and range data. Source: [5] 38

5.2 The inflated vehicle dimensions are crucial in analyzing the LIDAR point cloud for possible collision hazards to the vehicle. The collision circles are uniquely traced by the vehicle dimensions for each path about the ICC. 40

5.3 A sorting circle is traced by the inner, front corner of the vehicle as it turns about the ICC. Side and front obstacles are distinguished using the sorting circle that divides the area between the two collision circles. 40

5.4 All feasible paths must respect the R_{min} constraint of the vehicle. The paths begin at the baselink origin and end at a defined length along each path, l_{path}. 42
5.5 For each path in the set, the obstacles are projected onto the vehicle. This is analogous to assuming the vehicle will continue along the path and collide into the obstacles. The projection of the obstacles is useful for finding fpl. 43
5.6 The clearance to non-obstacles is calculated for each point in the point cloud. Only the minimum clearance value along each path is stored for use in the scoring function. 45
5.7 The distance between the endpoint of each path and the goal in the baselink frame is used as the distance to goal metric in the scoring function. 47
5.8 There are two expected velocity profiles of the OpenDani while driving under the 1D TOC implementation. 49
5.9 The OpenDani is placed on a chassis dynamometer. An open loop, voltage input test is performed to find the top speed of the vehicle. . 50
5.10 During a 1m test run, the OpenDani executes a trapezoidal velocity profile with a max velocity of 0.4 m/s. These results are a product of the logic used in Equation 5.15. 51

6.1 The level crossing at which $t_{latency}$ is evaluated is chosen as half the amplitude of the OpenDani velocity command signal. 56
6.2 The pre-mapped, collision area of a WMR's environment is inflated according to the joint distribution representing wheel speed uncertainties. Figure adapted from [6]. 60

6.3 The OpenDani's expected velocity profile for 1D-TOC and a given set of robot operating parameters is used to inspect the wheel speed controllers' performance. 62
6.4 Obstacles are placed in the OpenDani's operating environment. The OpenDani does not have prior knowledge of obstacle locations when executing obstacle avoidance. 67
6.5 The OpenDani is deployed into the obstacle ridden space with latency compensation and an appropriate safety bumper. 68
6.6 Latency compensation is disabled in the OpenDani system. Collision between the OpenDani and obstacles are observed. 69
6.7 The OpenDani dimensions are not inflated to compensate for wheel speed controller error. Latency compensation remains enabled. . . . 70
6.8 The OpenDani is operated without considering sources of motion uncertainty as Latency compensation and a safety bumper are removed from the obstacle avoidance algorithm. 70
6.9 The collision points on the OpenDani safety bumper trace circular, virtual paths as the vehicle turns about a ICC. These points are critical for safety analysis. 73
6.10 The resultant location of the vehicle's critical points, upon executing a critical stopping maneuver, are demonstrated for various turning radii and the straight driving case. 75
6.11 The traced path length of the outermost collision point is greatest at the minimum turning radius allowable. This trace length converges towards the straight driving trace length as the vehicle's turning radius increases. 75

6.12 For the R_{min} collision experiment, the test obstacle is placed in the path of the outermost collision point but not in the path of the other collision points. 80
6.13 The OpenDani collides with the obstacle when it's LIDAR range is set to $Range_{sub}$ for the straight driving case. 82
6.14 Though the OpenDani does not collide with the obstacle course when the LIDAR range is set to $Range_{min}$ for the R_{min} stopping case, the OA performance is degraded. 84
6.15 The OpenDani is able to maneuver throughout the obstacle course with improved performance when the LIDAR range is set to about double the minimum LIDAR range requirement for safe collision avoidance. 84

Chapter 1

Introduction

1.1 Objectives

The objective of this work is to introduce a low-cost, accessible design of a differential drive robot (DDR) research platform, the OpenDani. The improved design addresses shortcomings in a previous non-autonomous design to achieve elevated autonomy. Then, a safety analysis is conducted on the OpenDani while under a LIDAR-based obstacle avoidance functionality. The goal of this analysis is to relate software and hardware design choices to the safe operation of the robot.

1.2 Motivation and Justification

Autonomous mobile robots have become a popular research platform within the robotics field. This research is largely motivated by a push towards the mass deployment of autonomous vehicles (AVs). Companies such as Cruise, Tesla, and Waymo lead these efforts. It is projected that AVs may occupy 2% of light vehicle sales by 2030 [7]. For perspective, electric vehicles made up 1.1 % of light vehicle

sales in 2018 [7]. Additionally, small-scale autonomous mobile robots are in popular demand for a variety of applications such as dine-in delivery services, warehouse automation, and healthcare assistance, especially during the recent global COVID-19 pandemic [8, 9, 10]. As autonomous mobile systems increase their presence in society, it is crucial to understand the safe operation of these robots in the real world. Defining the safe operation of autonomous vehicles, especially in the vicinity of human-operated systems, continues to be an open and challenging research task. The process for defining the safe operation of mobile robots must be rooted in the analysis of the robot design itself as it pertains to expected functionalities of the robot within a given context (i.e. warehouse processing, on-road driving).

Determining safety during operation of a mobile robot relies on several components of the system. These components include the algorithms providing autonomy to the robot, the robot's sensing suite, the computational ability of the robot, and the physical constraints on robot movement. In examining some of these components, this thesis provides an analysis of safety for a mobile robot system during an obstacle avoidance task.

First, the Open Dani is built to serve as a low-cost platform capable of performing standard autonomous tasks. The wheeled mobile robot (WMR) design is compared to a previous, non-autonomous version, vDani. vDani primarily relied on overhead vision to operate. A review of the design choices will provide the necessary background knowledge for the safety analysis of the OpenDani. Additionally, OpenDani aims to maintain a low-cost design in the hopes that students and other researchers can have access to autonomous vehicle technology. The use of common, off-the-shelf components makes this low-cost design accessible. Upon building the new OpenDani design, a LIDAR-based obstacle avoidance (OA) algorithm is imple-

mented on the robot to verify the integration of software and hardware components.

Finally, the safety analysis of the OpenDani is founded in identifying sources of uncertainty during general robot operation. More specifically, motion uncertainty, environmental uncertainty, and perception uncertainty are examined to address safety concerns arising during OA. Human robot interaction or dynamic obstacles are not in scope for the safety analysis conducted in this work. Experiments are designed and conducted to highlight the safety concerns arising from these uncertainties.

An overview of the kinematic model and the assumptions adopted in using such a model for the OpenDani is provided in Chapter 2. The non-holonomic constraints of the OpenDani vehicle are also discussed within Chapter 2. In Chapter 3, a review of both the OpenDani and vDani designs is completed through analyzing the root autonomous capabilities of each vehicle version. The review gives context as to why a redesign of the older vDani version is warranted for this body of work. A Kalman-Filtering method is developed to process wheel speed signals from the low resolution encoders used in the OpenDani system. This is an obvious byproduct of a low-cost robot design. The low-level controller and wheel speed filtering formulation are elaborated on in Chapter 4. In Chapter 5, the OA scoring function and point cloud filtering methods are discussed as it is later found that these factors relate to the minimum LIDAR Range requirement of the OpenDani vehicle. Finally, the OpenDani is deployed in an obstacle course environment. The results from a series of collision experiments are presented in Chapter 6. The findings of this work are concluded in Chapter 7, alongside a proposal for future work.

Chapter 2

OpenDani Model Overview

2.1 Introduction

Wheeled mobile robots (WMR) are best described by their drive system type. Amongst the most popular drive systems are Ackermann steering, differential drive, and omnidirectional. The drive system is governed mostly by wheel type and the number of actuated or free wheels belonging to the WMR. A review of mobile robots highlights WMRs as a low-cost and practical solution for tasks performed on relatively level surfaces [11]. The OpenDani is expected to function in a flat, indoor environment. Hence, a WMR design was adopted for this work.

A differential drive system (DDS) is best used to describe the OpenDani WMR. Two separate DC motors are used to actuate wheels fixed at each side of the robot base. A tracking wheel, which enables two degrees-of-freedom (2-DOF) motion, is placed at the rear end of the robot base to balance the system. The 2-DOF tracking wheel assists in avoiding scrubbing motions when performing turns. Disadvantages that arise in a DDS include driving error introduced by actuator differences. This error is most evident when driving in a straight line. Sufficient

tuning of the actuators' low-level controller gains are required to mitigate this issue. An advantage in choosing a DDS is that a three-wheeled system does not require a suspension system. This further enables the OpenDani in maintaining a low-cost design.

2.2 Differential Drive System Model

The OpenDani can be sufficiently modeled using a *kinematic* model. The popular kinematic model introduced in [12] is adopted to describe the position and orientation of the OpenDani with respect to an inertial reference frame. The set of coordinates x, y, ϕ are applied to the system as reflected in Figure 2.1 below:

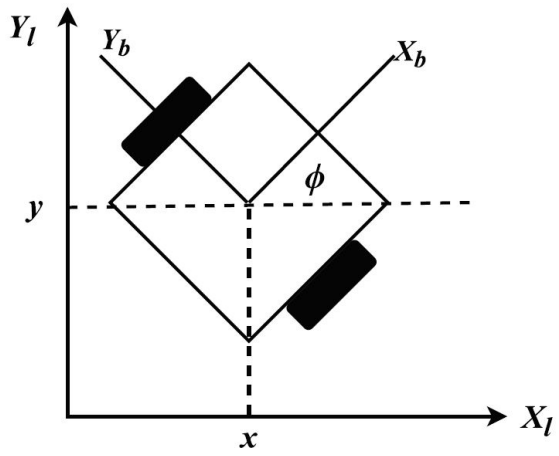

Figure 2.1: A reference frame and coordinate representation for a differential drive robot. Source:[1]

A coordinate frame fixed on the robot body is referred to as the *baselink* frame. The origin of the baselink frame is placed in between the drive wheels, at the intersection of the X_b-axis and Y_b-axis illustrated in Fig. 2.1. The matrix formulation in Equation 2.1 summarizes the kinematic model representing the WMR

configuration. The position of the OpenDani is represented by x and y. The heading angle or orientation of the robot is described by ϕ. Inputs into the system model are the linear and angular velocities desired of the robot:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.1}$$

In adopting a kinematic model, forces and torques acting on the robot body are not considered in deriving the robot's configuration from a control input $[v, \omega]$. The following assumptions were made in using a kinematic model:

(1) Wheel slip is negligible.

(2) The mobile robot traverses on a perfectly flat plane. Motion in the z-direction is non-existent.

(3) The inertia of the robot is insignificant when predicting the OpenDani's resultant configuration given a control input.

A discrete form of the kinematic model is used to develop an odometry formulation for the OpenDani. The position and orientation of the WMR is estimated using the set of Equations 2.2:

$$\begin{aligned} x_{k+1} &= x_k + v\cos\phi_k dt \\ y_{k+1} &= y_k + v\sin\phi_k dt \\ \phi_{k+1} &= \omega dt \end{aligned} \tag{2.2}$$

By acquiring wheel encoder feedback and employing a dead reckoning formulation, the odometry efficiently estimates the configuration of the OpenDani for short dis-

tances. The time step for which the robot forward and angular velocities, v and ω, are updated is reflected by the dt term in the Eqn. 2.2 above. By incrementally updating estimated v and ω, odometry drift affects can be minimised[13].

Dynamic models have also been used to model DDSs. One model considers wheel traction forces to avoid wheel slip in a DDS by limiting the control inputs accordingly [14]. A dynamic model of a DDS is obtained using a Lagrange Formulation and is combined with a DC Motor model to achieve a power consumption model of the system [15]. Energy-based methods that examine the DDS dynamics in its electrical and mechanical domains are used to develop an energy model that closely approximates the energy consumption of the DDS [16].

The OpenDani is expected to experience low velocities. The maximum robot forward velocity is is 1 m/s. The maximum acceleration and deceleration of the robot is 1 m/s^2. For experiments presented in this work, the robot velocities and accelerations are typically set below the aforementioned maximum values. When operating at these conditions, it is reasonable to assume that wheel slip and robot inertia will not cause significant misrepresentation by the kinematic model.

2.3 OpenDani Wheel Velocity Mappings

The linear and angular velocities of the robot, v and ω, have been introduced as control inputs into the OpenDani system. These control inputs must be further deconstructed when developing the low-level actuator controllers. A set of v and ω inputs are achieved by independently driving the OpenDani's wheels at unique angular velocities. The angular wheel velocities, ω_l and ω_r, for a set of control inputs $[v, \omega]$ can be found using the following relationships:

$$
\begin{aligned}
R &= \frac{v}{w} \\
w_l &= w\frac{R - \frac{l}{2}}{r} \\
w_r &= w\frac{R + \frac{l}{2}}{r}
\end{aligned}
\tag{2.3}
$$

R shall be referred to as the *turning radius* of the robot. This is the distance from the origin of the OpenDani's baselink frame to the instantaneous center of curvature (ICC) of the vehicle. R can be given as a Y_b-coordinate in the baselink frame. The wheel radii is represented by r. l is the width of the OpenDani body measured from the left to the right wheel.

For better human legibility, the desired forward velocity of the robot v is set. Then, the turning radius R is given as input rather than angular velocity, ω. The input set $[v, \omega]$ is then better represented as $[v, \frac{v}{R}]$.

An exception to the wheel velocity mappings presented above arises when the robot moves without an angular velocity, ω. Such is the case when the robot moves in a straight line and R is infinitely large. The independent wheel velocities for the straight-line case are given as:

$$
\begin{aligned}
\omega_l &= \frac{v}{2r} \\
\omega_r &= \frac{v}{2r}
\end{aligned}
\tag{2.4}
$$

Note that in this work, only forward wheel velocities are commanded. The OpenDani is not driven backwards in the present study. Hence, the robot is not able to turn in place, about it's own center. This type of 'zero-radius turn' movement can be an advantage of a DDS. Wheel velocities must be equal in magnitude but opposite in direction to perform this maneuver. Yet, there are constraints on the robot velocities that must be identified.

2.4 Configuration Space and Non-Holonomic Constraints

A configuration space (C-space) can be used to denote all the possible configurations of a rigid body system. This is especially useful for robot motion planning in the presence of obstacles [17]. OpenDani can be represented as a rigid body that has a C-space denoted using the coordinates x,y, and ϕ as applied to Figure 2.2 below. The C-space of the OpenDani, described herein, is of a rigid body rotating and translating in a 2D-plane. In the absence of obstacles, the C-space spans $x \in \mathbb{R}$, $y \in \mathbb{R}$, and $\phi \in [0, 2\pi]$. The entire C-space of OpenDani is denoted as [2]:

$$C = \mathbb{R}^2 \times [0, 2\pi] = \mathbb{R}^2 \times S^1 \tag{2.5}$$

This C-space is also referred to as the special Euclidean group $SE(2)$. The C-space representation above is invalid when obstacles exist in the robot's plane. Still, the C-space is useful in analyzing motion constraints on a rigid body. In addition to holonomic constraints, such as the one that constrains OpenDani's motion to a plane, there is non-holonomic constraints on the system. Non-holonomic constraints

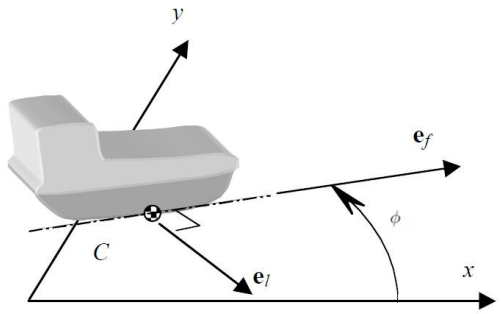

Figure 2.2: A rigid body with constrained movement to a plane. Source:[2]

act on the velocities of a rigid body. Note the e_i-axis, shown in Fig. 2.2. For the

OpenDani, lateral movement along the e_i-axis is restricted. The OpenDani's wheels allow the robot to roll forward, however, it is unable to slide laterally to reach a point directly to the left or right of it's body. This non-holonomic constraint is known as the knife-edge constraint [2]. It is mathematically desribed as:

$$\dot{x} \sin \phi - \dot{y} \cos \phi = 0 \tag{2.6}$$

In the context of safety, it is important to note that the OpenDani cannot slide laterally for the purposes of efficient obstacle avoidance. Instead, the OpenDani can either come to a stop or detour by turning away from an obstacle. The minimum turning radius, R_{min}, is a critical robot motion characteristic imposed by the non-holonomic constraint. As previously mentioned in Chapter 1, the OpenDani is not able to execute negative angular wheel velocities to achieve $R_{min} = 0$. The R_{min} of the OpenDani is regulated by the low-cost actuators. Due to stiction present at low angular velocities of the DC motor actuators, the OpenDani has an R_{min} of 0.7 m. The R_{min} of the robot is a critical characteristic that is used to define the safe operation of the robot during the OA task.

Chapter 3

Autonomous Robot Design

3.1 Introduction

A WMR serves as a research platform to achieve the objectives presented in Chapter 1 of this thesis. A principal contribution of this work was the development of OpenDani, a low-cost WMR that can be used to develop and implement autonomous navigation algorithms. When designing the OpenDani, sensors and software components were carefully chosen to support autonomy. A design overview for the OpenDani will elaborate on how these components can support state-of-the-art autonomy. In contrast to the commercial Dani system (NI, Inc., Austin, Texas), the OpenDani has adopted a modular design using an on-board computer. It can easily interface with a variety of sensors and open source software packages.

A review of the previous vDani robot highlights shortcomings of the system in qualifying it as an autonomous system. The absence of on-board computing tethered the vDani design to a desktop computer, severely limiting portability. vDani relied on an external vision system for feedback into the motion control loop. This limited the robot's workspace, making it difficult to execute medium to long range tasks.

The vDani executed only a pre-mapped, *offline* obstacle avoidance method.

Both the vDani and OpenDani rely on a Robot Operation System (ROS) software framework. The NI Dani does not use ROS by default and instead uses proprietary software (i.e. LabView). ROS allows for the easy data communication among software nodes in a robotic system [18]. Off-the-shelf, open-source software packages such as the mapping package *gmapping* and the global planning package *global_planner* are available for free download [19, 20]. Through ROS, OpenDani can maintain a relatively low-cost software development approach when new functionalities need to be tested within a short time frame.

3.2 Analyzing Autonomy

The evaluation of mobile robot autonomy in the proceeding sections has established the need to define an autonomous robot. For AVs, the National Highway Traffic Safety Administration and the Society of Automotive Engineers have defined vehicle automation on a level basis (0-5) in relation to human driver intervention [3]. The levels of autonomy are summarized in Figure 3.1.

Though the amount of human intervention in a robotic system can provide an abstract indication of it's autonomy, a more concrete definition of autonomy is necessary to analyze the WMR design. In a performance study evaluating mobile robot autonomy during navigation tasks, the agent under investigation is given mapping, localization, planning, obstacle avoidance, and general supervision (perception) capabilities [21]. The Autonomy Levels for Unmanned Systems Framework (ALFUS) authored by the National Institute of Standards and Technologies (NIST) defines the Root Autonomous Capabilities (RAC) of an unmanned system (UMS) to be sensing, perceiving, analyzing, communicating, decision-making, and acting/executing [22].

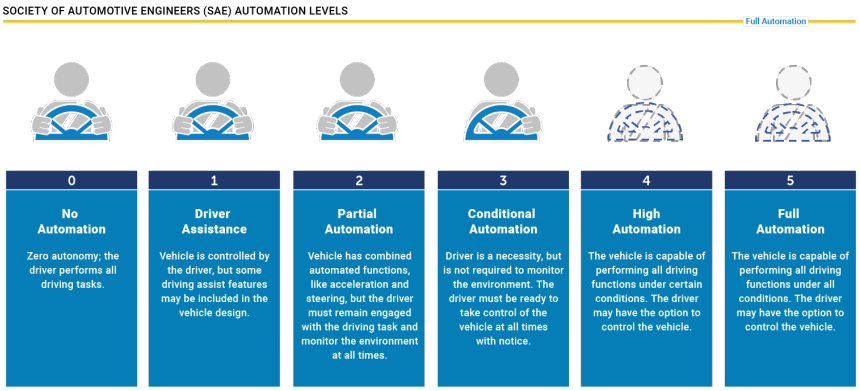

Figure 3.1: Autonomy levels for an AV are based on the amount of human driver intervention necessary to operate the vehicle. Level 5 autonomy requires no human intervention under all conditions to perform all driving functions. Source: [3]

Taking the RAC and the work in [21] into consideration, the vDani and OpenDani designs are evaluated in their ability to support *self-localization*, *perception*, *motion planning*, and *trajectory tracking*.

3.3 Review of Previous vDani System

In a previous study, the vDani (see Figure 3.2) was designed to perform a box pushing task [4]. The robot's hardware includes a metal chassis, a rubber bumper to interface with objects, two drive wheels, and a 2-DOF tracking wheel. An AprilTag is attached to the robot body. It is detected by a computer-vision based, tag recognition system [23, 24]. The electronics on the vDani include two DC motors used for actuation, two rotary encoders connected at each motor shaft, and a NI MyRIO (NI, Inc., Austin, TX) real-time board running the low-level control loop for both actuators. All electronics are powered by a single 12-volt RC battery unit.

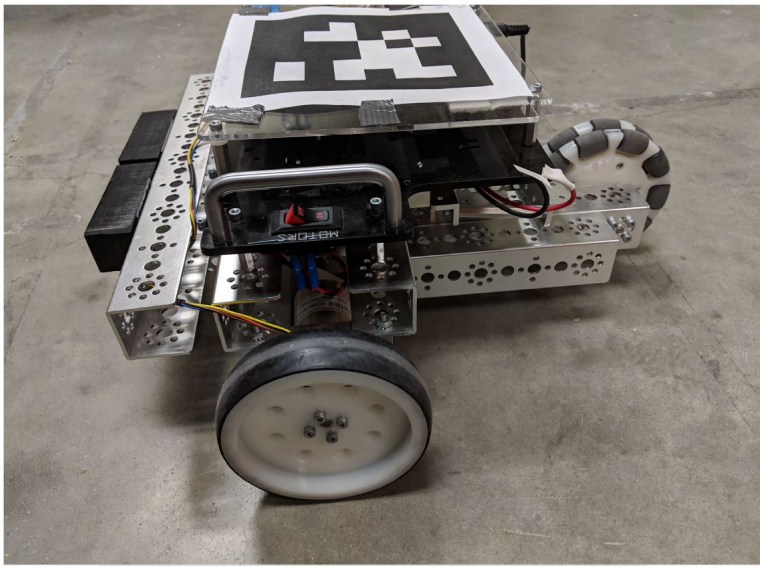

Figure 3.2: The previous robot design, vDani, contains only a paper AprilTag atop it's body. Source: [4]

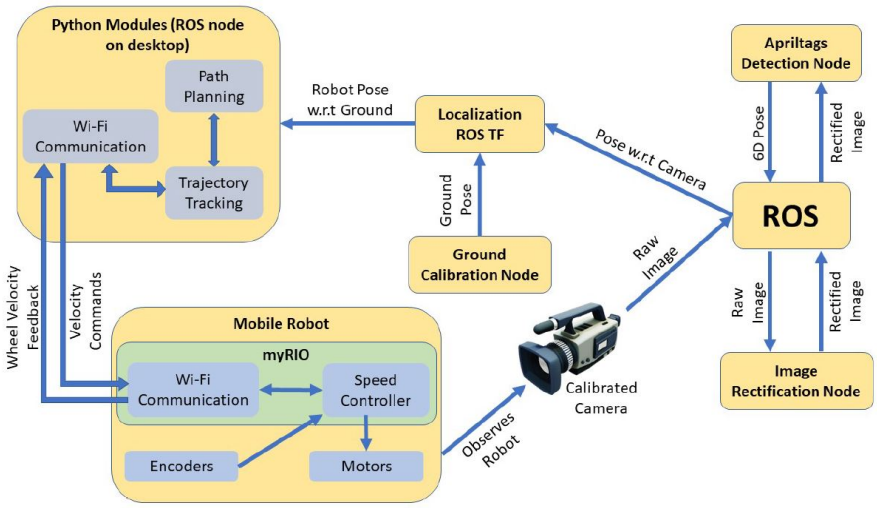

Figure 3.3: The system architecture of the vDani system relies on the communication between various ROS nodes. A camera routes images into an AprilTag detection node using the ROS communication network. Source: [4]

The diagram in Figure 3.3 illustrates the system architecture, showing how the vDani was integrated with ROS to use a variety of software nodes. To complete a box-pushing task, the constrained navigation task involves moving a small box from point A to B. The robot sequence is described from Figure 3.3:

1. With the box in contact with the robot bumper, a trajectory path is generated *offline* within the path planning node using MATLAB. The path is manually transferred to the trajectory tracking node. Both nodes are run on the desktop computer.

2. The camera-based, AprilTag recogntion vision system provides the configuration $[x, y, \phi]$ of the vDani within a camera image frame.

3. The error between the vDani's detected configuration and the configuration given by the path planner is calculated. The error in the x, y, and ϕ coordinates are used for a P controller employed by the trajectory tracking node.

4. $[v, w]$ commands from the P controller are sent to the MyRIO board over the wifi network. The low-level speed controller on the MyRIO executes the necessary wheel velocities for path following.

Many of the vDani system's limitations are rooted in its sensing abilities. Moreover, the vDani system does not capitalize on all available sensor information. For example, wheel velocities are logged to a file for post-processing but are not used as feedback into the robot system. Lastly, the vDani utilizes the NI myRIO computation device which does not easily interface with low-cost, plug and play sensors or open-source software.

3.3.1 Self-Localization

An AprilTag is attached to the vDani body for localization purposes. The AprilTag recognition software determines the robot's pose with respect to the camera frame [23, 24]. This localization method imposes a significant limitation on the robot's workspace. The camera's image frame only covers about a 4 m^2 section of the ground that vDani traverses upon. vDani may only be localized within this 4 m^2 space before information about the robot's location is completely lost.

The AprilTag localization method may be sufficient for circumstances where a robot is fixed to it's environment. Such is the case for the improvement of manipulator position accuracy on the Mars Rover prototype presented in [25]. A camera captures the fiduciary tag on the manipulator's end-effector to localize the end-effector. A mobile robot should be able to traverse for some distance. This method of localization is not practical for even mid-range tasks of a mobile robot. Because the localization task for the vDani is performed by a separate entity, the AprilTag recognition system, the lack of portability of the vDani robot is evident once again. If the vDani is moved to another location, the vision system must accompany it.

3.3.2 Perception

The vDani system's perception is limited to feedback from the vision system. The vision system is only able to recognize AprilTags within the camera frame. All objects without AprilTags are incapable of being perceived by the vision system and, therefore, by the vDani. It is not possible to identify untagged, unmapped objects with the AprilTag vision system alone. The addition of an object detection software would be necessary to find objects in the images captured by either the external vision system or an on-board camera. A moving object detection algorithm tracks

sets of feature points over consecutive image frames with a monocular camera [26].

3.3.3 Planning

The vDani obtains a motion plan to maneuver a virtual world with *known*, fixed obstacles from an *offline*, sample-based path planner. Sample-based planners sample a configuration space and construct a road map from these samples. The vDani planner is based on the Rapidly-exploring Random Trees (RRT) algorithm [27]. Since planning is completed in MATLAB, it is necessary to manually transfer path way points into the ROS network. The time to plan for the vDani in MATLAB, given a virtual world containing two obstacles, is about two minutes. A human-in-the-loop is necessary to place the path way points into the trajectory tracking node. The lack of a ROS-based planning node disrupts the ability to efficiently plan and execute a path. Additionally, due to the lack of perception by the vDani, replanning to maneuver around unmapped obstacles is not possible.

3.3.4 Trajectory Tracking

Given the path generated by the RRT planner, no explicit time constraints are placed on the vDani in reaching the goal way points. It is best stated that the vDani system then executes *path following* not *trajectory tracking* to traverse along a path. A simple P controller is used to successfully implement path following [4]:

$$\begin{aligned} v &= k_1 x_e \\ w &= k_2 y_e + k_3 \phi_e \end{aligned} \tag{3.1}$$

3.4 Improvements to the System

To extend and improve upon the existing capabilities of the vDani, design changes are focused on improving support for self-localization, perception, and planning techniques. Design changes include increasing sensing capabilities, augmenting on-board computation, and revamping software implementation. Table 3.1 summarizes the changes made to various components of the vDani system.

Components	vDani	OpenDani
Low-Level Actuator Control	MyRIO Real-Time Board	SparkFun RedBoard (Arduino)
CPU	Desktop PC: Intel Core i5 Processor	On-Board Raspberry Pi 4B: Quad-core Cortex-A72 Processor
Operating System	Linux Ubuntu 16.04	Linux Ubuntu 20.01 (headless)
Software Middleware	ROS Kinetic	ROS Noetic
Sensors	Rotary Encoders (400 ticks/rev)	Rotary Encoders (200 ticks/rev)
		Slamtec RPLIDAR-A1
		USB Camera (Optional)
Power	12 Volt RC Battery	12 Volt RC Battery

Table 3.1: The vDani and OpenDani design mainly differ in their sensor suite and computational specifications. The OpenDani's Raspberry Pi contains a ROS installation that interfaces well with open-source software.

An on-board, Raspberry Pi(RPi) 4B single-board computer is mounted on the OpenDani. The RPi hosts the entire ROS network. Most notably, the RPi easily interfaces with a LIDAR sensor and a USB camera via it's USB ports. The LIDAR sensor is permanently mounted on the OpenDani body. An off-the-shelf, ROS software package can interface with most USB cameras to route images into the ROS network[28]. The AprilTag vision system is eliminated from the motion control loop. The OpenDani contains a C++ and Python software implementation. Motion algorithms are implemented entirely in C++. Python was used to develop

the ROS nodes that handle communication between the RedBoard MicroController and RPi.

3.5 Cost Overview

Maintaining a low-cost was a major design constraint in the redesign of the vDani system. In building a low-cost mobile robot, the platform becomes more accessible to other researchers or STEM educators. Cost is a main factor in the success of teaching robotics concepts throughout varying levels of education [29]. OpenDani uses off-the-shelf components and open-source software that allows similar designs to be rebuilt at home, in a lab, or in the classroom setting. Table 3.2 shows the base cost of components shared by both the vDani and OpenDani.

Base Cost	
Aluminum Chassis	~$80
2 Drive Wheels	~$20
1 Tracking Wheel	~$20
2 DC Motor & Mounts	~$40
12 V NiMH Battery	~$50
Modular, Rotary Encoders	~$80
Total	~$290

Table 3.2: The vDani and OpenDani share several hardware components. These shared components make-up a base-cost for the two designs.

The base cost of both systems is composed mostly of hardware costs. The total base cost is about $290 dollars. The most affordable robotic learning platforms are between $100-$150. The Spiderino, a toy-robot that supports swarm robotics concepts, is an example of a learning platform within this price range [30].

The vDani and OpenDani have additional costs unique to their design. Details of these costs for each design are listed in Table 3.3. The base cost for both

Additional Costs		
	vDani	**OpenDani**
Microcontroller	~$500	~$20
CPU	~$500	~$98
LIDAR	N/A	~$100
Camera	N/A	~$30
Total	~$1,000	~$248

Table 3.3: As the robot designs deviate from each other, so does their total design cost. The additional cost of each robot summarizes the cost in design differences.

designs surpasses the costs of the most affordable robotic learning platforms. However, the OpenDani's total cost of about $540 is below the cost of a similar platform, MuSHR [31]. MuSHR is a low-cost robotic race car with a cost of about $900. It has similar components and expected functionalities to those of OpenDani.

3.6 Overview of the OpenDani System

The OpenDani repurposes the DC motors and metal chassis of the vDani system[1]. The OpenDani is a product of new software implementations and the adoption of modern electronics and sensors. OpenDani can actively avoid unmapped, static obstacles. Due to non-holonomic constraints, OpenDani stops to avoid collisions during instances when detouring is not possible. Online obstacle avoidance is made possible through a new odometry node integrated into the ROS network. A point cloud is built from OpenDani's LIDAR sensor data. The point cloud is analyzed to make intelligent decisions using a greedy, obstacle avoidance algorithm. The OpenDani prototype, pictured in Figure 3.4, is visibly different from the vDani prototype. The OpenDani is entirely self-contained and capable of autonomous driving tasks.

[1]The original NI Dani used components available from pitsco.com.

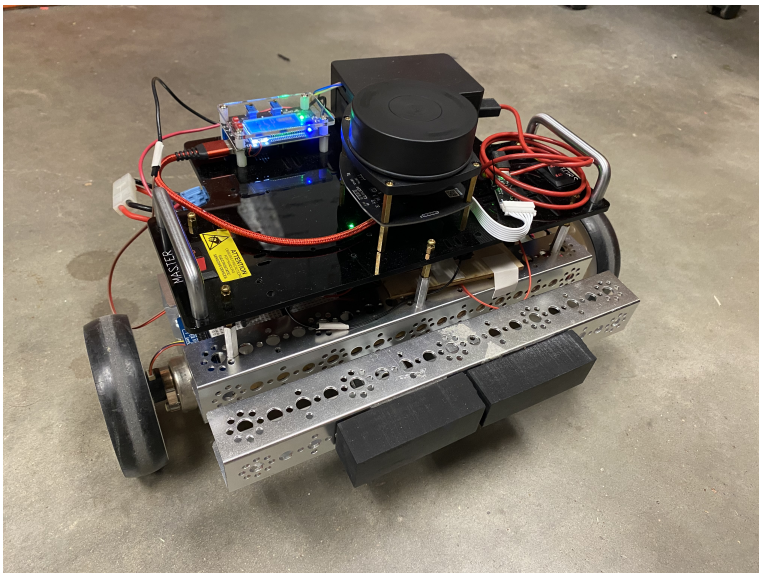

Figure 3.4: The OpenDani has several components mounted atop it's body. The components include a LIDAR sensor, a single-board computer, and a DC buck converter. A RedBoard micro-controller is not visible.

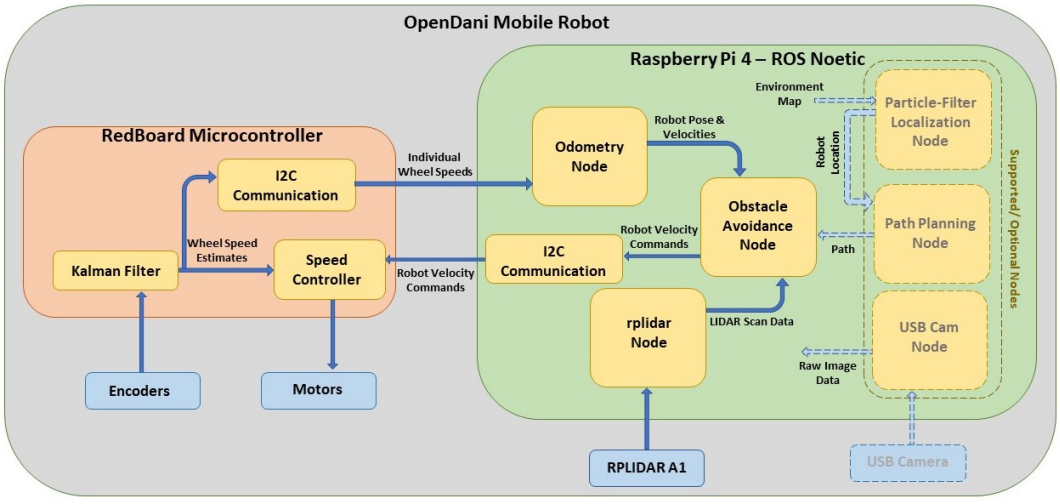

Figure 3.5: A map of the OpenDani system architecture depicts the data transfer between software nodes for an obstacle avoidance functionality.

3.6.1 Self-Localization

Unlike vDani, the OpenDani is capable of being localized outside of a single camera's image frame. Self-localization abilities of the OpenDani are supported through the acquisition of a LIDAR sensor and the implementation of an odometry node. Mobile robot localization schemes are available for both outdoor and indoor applications. Outdoor localization methods typically utilize GPS and IMU sensors. An outdoor, EKF-based localization algorithm in [32] uses a slip-proof optical sensor, GPS, and IMU to estimate the location of a skid-steer robot on uneven ground. OpenDani has been developed for indoor applications. A suitable localization scheme for OpenDani may be obtained using Monte-Carlo (MC) localization [33]. This localization method requires a reference map of the indoor environment. Real-time LIDAR readings must be compared to the map to develop an observation-likelihood model that provides insight into the possible locations of the robot within the map. This information is leveraged in the update step of the MC localization algorithm. The prediction step of the algorithm is odometry-based and samples possible robot locations from a gaussian noise model [33]. The prediction estimates the location of the robot after some movement.

Other localization methods include camera-based algorithms such as Orb-SLAM [34]. OrbSLAM uses a feature-based extraction algorithm to localize the agent over a series of images while simultaneously rebuilding the environment.

3.6.2 Perception

Previously, the vDani robot did not have it's own method to perceive its environment. Instead, an external vision system was used to perceive all tagged objects such as the vDani itself and a small box. The OpenDani can perceive its environ-

ment using a LIDAR sensor. The LIDAR has a field of view (FOV) of 360° and a maximum sensing range of 12 m. The LIDAR sensor is suitable for implementing environment rebuilding algorithms. That is, by using LIDAR scan data, a map of the environment can be built via correlative scan matching (CSM) [35]. CSM stores a single LIDAR scan after each sampling period so that a comparison to a subsequent scan can be made. A rigid body transform between the two scans is found by maximizing the probability that the two scans have observed the same features. The rigid body transform is applied to scan data using a predicted robot location to reconstruct the environment.

The ability to rebuild a map of a robot's environment is an indication of the perception capabilities of a robot. Furthermore, mapping is crucial when digitized blueprints of an environment are not available or if the environment significantly deviates from map archives due to presence of un-mapped furniture or fixtures. A simultaneous localization and mapping implementation (SLAM) addresses both localization and mapping in parallel. In solving the SLAM problem, a robot becomes aware of its own location within the environment it is exploring and reconstructing. For the OpenDani, a practical application of SLAM is the pose-graph slam [36].

Additional modes of perception can be achieved using monocular, stereo, or depth cameras. The OpenDani's on-board RPi provides a level of modularity with it's plug-and-play interface to USB sensors, such as cameras. An off-the-shelf ROS camera node seamlessly routes camera image data into the OpenDani system [28]. This allows for future implementations of image-based algorithms for localization, mapping, or other computer vision applications on the OpenDani design.

3.6.3 Planning

Similar to vDani, the OpenDani has the ability to plan when given a map of an environment. This map can come in the form of digitized blueprints or from the implementation of OpenDani's own map building solution. For vDani, the search space is a discretized C-space for all $[x, y, \phi]$. Depending on the discretization resolution and the dimensions of the real-world represented by the C-space, building the 3D search space could take a significant amount of computation. Instead, it is better to use a lattice-based search-space approach presented in [37]. This method generates a 2D search-grid by considering the motion constraints of a WMR. The nodes of the graph are connected by edges that satisfy kinematic-ally feasible paths. Kinematic-ally feasible paths are paths that do not violate the R_{min} constraint of the OpenDani. By using standard search algorithms, such as the A* algorithm [38], an optimal path along a lattice-based grid can be found. The A* algorithm aims to minimize the cost of a path during search. The cost of a path is defined by a base cost and a user defined heuristic cost. The distance traveled between graph nodes is typically set as the base cost. An example of a heuristic cost is a node's distance to the end goal.

For the vDani, planning was performed in MATLAB. OpenDani's core software language has been chosen to be C++ due to it's popularity in robotic applications. A study has shown MATLAB to be up to 10 times slower than C++ in solving complex algorithms such as value iteration [39]. By writing the high-Level OpenDani software using a C++ ROS node implementation, planning methods such as A* search execute faster. Most importantly, the manual transfer of path data to the trajectory following node is no longer necessary. This eliminates the necessity for a human-in-the-loop that is necessary in the vDani system.

3.6.4 Trajectory Tracking

Given a series of way points from the path planner, the OpenDani must be able to traverse along the defined path. Trajectory tracking is concerned with both spatial and temporal constraints along a motion plan. Path following methods are often suitable for mobile tasks that are not under strict time constraints. Pure pursuit is a general path following algorithm that calculates the arc to a goal point [40]. To execute path following, control inputs are generated for a WMR so that it traverses the set of arcs calculated between a series of way points. A simplified variation of the pure pursuit algorithm can be integrated with the obstacle avoidance method that has been implemented on the OpenDani. Pure pursuit can be used in-place of the closed loop P-control used for path following in the vDani system.

Chapter 4

Low-level Control

4.1 Introduction

The OpenDani control hardware uses low-cost and easily accessible components. Figure 4.1 shows the power transfer and data communication between all low-level, electronic components. High-level commands from motion algorithms running on the RPi Linux computer are transferred to a RedBoard micro-controller via an I2C communication protocol. The RedBoard maps these high-level commands to wheel speeds using the relationships discussed in Chapter 1. The appropriate pulse-width modulation (PWM) signals are sent from the RedBoard to the motor driver for motor speed control. Two external interrupt pins on the RedBoard are interfaced with two modular shaft encoders to sense wheel rotation.

For wheel speed control, a model-free PID controller is employed using wheel speed feedback from the shaft encoders. A byproduct of the low-cost OpenDani design is a noisy speed signal from the low-resolution encoders. To improve the noisy speed signal, a Kalman filter approach was applied to acquire a refined wheel speed estimate. This method is compared to the classic, finite-difference approach

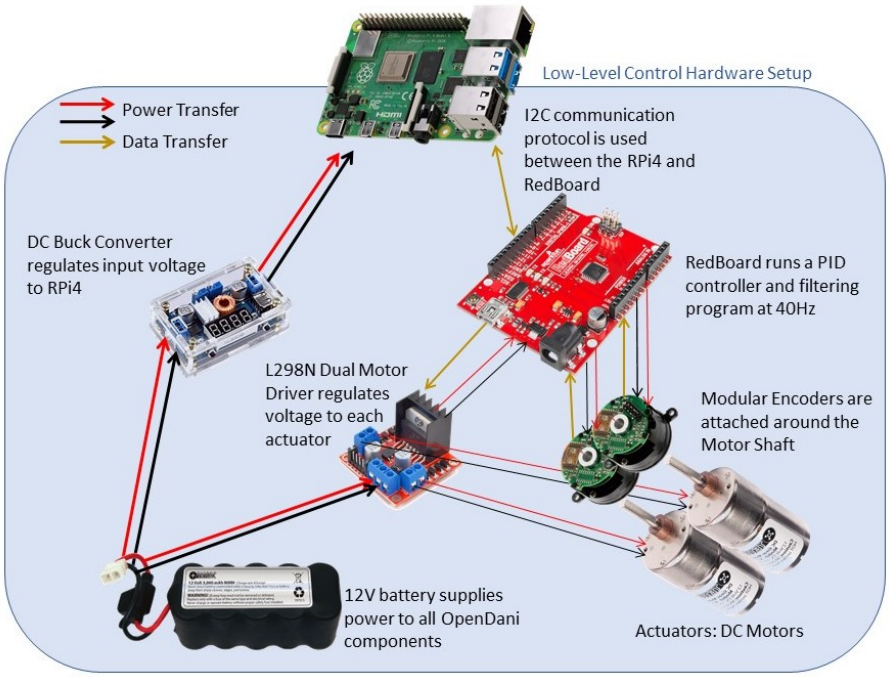

Figure 4.1: A diagram shows the power transfer and data transfer between the low-level control components. An I2C communication protocol bridges the communication gap between the OpenDani Linux computer and the RedBoard micro-controller.

used for estimating angular speed via shaft encoders.

4.2 Finite-Difference Estimation for Wheel Speed

A typical method for calculating angular shaft speed of a DC motor is to program the micro-controller's interrupt pins to count encoder pulses, n_{pulses}, between a sampling period, T_s. Then, taking a constant speed assumption between sampling periods, a finite-difference formulation provides the angular velocity of the motor shaft or the wheel speed, w_{wheel}:

$$w_{wheel} = 2\pi \frac{n_{pulses}}{PPR \times T_s} \quad \text{rad/sec} \tag{4.1}$$

In Equation 4.1, PPR is the known pulse-per-revolution value for the wheel encoders. A simple conversion is applied to obtain wheel speeds in revolution-per-minutes (rpm).

Given a step input, the wheel speed estimate using the finite-difference method is significantly noisy. In Figure 4.2, the speed estimate oscillates between 72 rpm and 84 rpm as the wheel velocity reaches a steady state. This noise can be attributed to the quantization present from low-resolution wheel encoders.

Consider a single pulse difference in the n_{pulses} from two sequential encoder count observations taken at steady state. The noise in the speed estimate caused by the single pulse difference is the quantization step size represented by:

$$\Delta w_{quant} = 2\pi \frac{1}{PPR \times T_s} \quad \text{rad/sec} \tag{4.2}$$

Note that each pulse accounts for a quantization step of Δw_{quant} in the speed estimate. A Δw_{quant} of about 1.25 rad/s or 11.94 rpm is found for the OpenDani

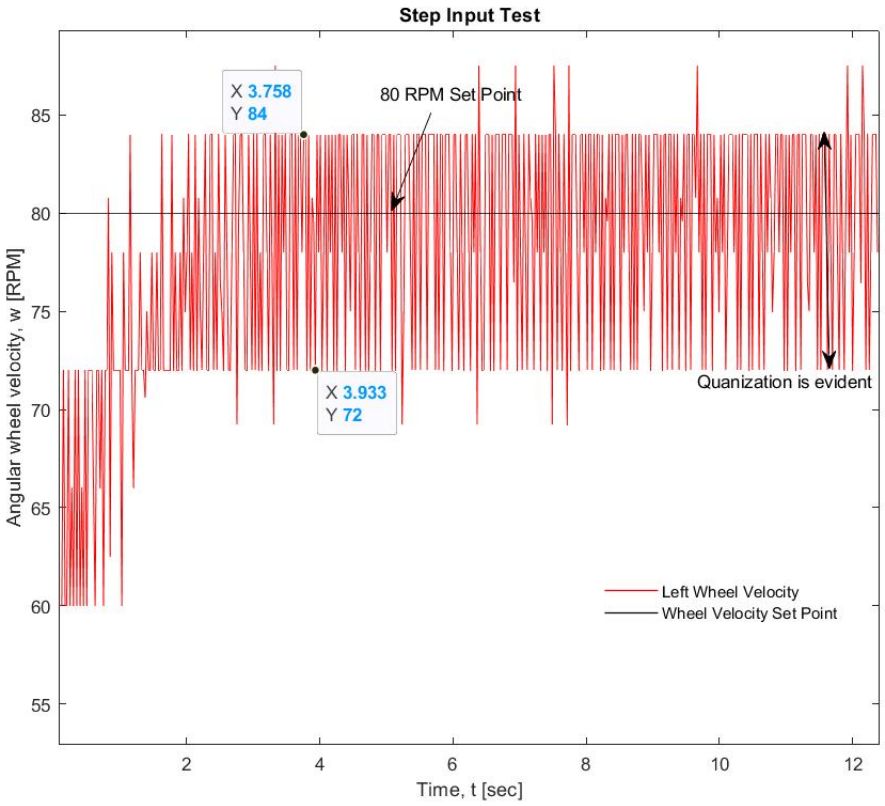

Figure 4.2: A simple step input test is performed on one of the OpenDani wheels. Upon reaching a steady-state condition, quantization noise is observed around the set point speed.

encoders. These encoders have a 200 PPR value and a T_s of 25 milliseconds. The calculated Δw_{quant} value agrees with the quantization noise highlighted in Fig. 4.2 and explains the oscillatory behavior around the set point. For shorter sampling periods, the quantization step size is expected to increase. Filtering is neccesary to deal with quantization noise.

4.3 Kalman-Filter Theory

A Kalman filter (KF) is a set of mathematical formulations that can reasonably estimate the state(s) of a system under modeling errors and noisy system measurements [41]. The KF can be formulated as an optimization problem with the objective of minimizing the mean-squared error between the state estimate and the true value of the state [42].

With an initial estimate of the system state, the KF performs a prediction step to estimate the next state, $\hat{x}'_{k+1}$. The initial state estimate, $\hat{x}_k$, is propagated forwarded in time using a state transition matrix, Φ_{st}. The state transition matrix is derived from a system model of the process. The prediction step for the recursive KF is formulated as such:

$$\hat{x}'_{k+1} = \Phi_{st}\hat{x}_k + w_k \tag{4.3}$$

Disturbances in the system can be captured by the process noise random variable, w_k. Furthermore, knowledge of the uncertainty in the initial estimate must be provided to the filter. This is known as the error covariance matrix, P_k, and it is also updated during the prediction step to reflect uncertainty in each state estimate. If the initial state estimate provided to the filter is known to be the true state, then the diagonal values of P_0 are set to 0 to reflect zero uncertainty in the initial state

estimate, $\hat{x}_0$. The error covariance matrix is updated in the prediction step:

$$P_{k+1}^{'} = \Phi_{st} P_k \Phi_{st}^T + Q \tag{4.4}$$

The Q matrix in Equation 4.4 above is referred to as the process noise covariance matrix. The values in Q reflect the noise introduced into the state estimates from modeling errors.

Next, the update steps of the KF are executed. By accounting for the predicted error covariance matrix, $P_k^{'}$, and the measurement noise covariance, R, an optimal Kalman gain is calculated to represent the confidence in the state measurement,

$$K_{gain} = P_k^{'} H^T (H P_k^{'} H^T + R)^{-1} \tag{4.5}$$

where H is the state measurement matrix. The Kalman gain is a value between 0 and 1. A value close to 1 indicates a high confidence in the state measurement while a value closer to 0 indicates a low confidence. The KF produces a state estimate that is mostly model derived when the Kalman gain is closer to 0. The Kalman gain is also influenced by R. The values in R are best set to reflect sensor noise metrics provided by a manufacturer.

Finally, the two equations in the update step give the final state estimate, $\hat{x}_k$, and a final error covariance estimate, $\hat{P}_k$. These final estimates are available upon processing a state measurement, z_k. The Kalman gain found using Equation 4.5 is used in the final update step,

$$\hat{x}_k = \hat{x}_k^{'} + K_{gain}(z_k - H\hat{x}_k^{'}) \tag{4.6}$$

$$\hat{P}_k = (I - K_{gain} H)\hat{P}_k^{'} \tag{4.7}$$

In Equation 4.7 above, I is an identity matrix. A discrete, recursive KF is implemented on the RedBoard to improve wheel speed estimates. The recursive formulation of the KF is practical for implementation on low-memory devices since only the previous $\hat{x}_k$ and $\hat{P}_k$ values must be stored.

4.4 Kalman Filter Based Wheel Speed Estimation

To properly estimate wheel speeds, a model is required for the prediction steps of the KF. Similar to the work in [43], a discrete-time, all-integrator model is adopted to describe the wheel system:

$$\begin{bmatrix} \hat{\theta}_{k+1} \\ \hat{\omega}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{\theta}_k \\ \hat{\omega}_k \end{bmatrix} \tag{4.8}$$

The motor shaft position, $\hat{\theta}_k$, and the angular shaft velocity, $\hat{\omega}_k$, are propagated forward in time with a time step, T_s. The model assumes a constant angular shaft velocity between sampling periods. This introduces some process noise to the system since this assumption is not always true. It is expected that the KF can compensate for the modeling error. The measurement taken from the system is represented by,

$$z_k = H\hat{x}_k + v_k \tag{4.9}$$

The measurement matrix, H, indicates a measurement of the shaft angular velocity given by encoders at a fixed-time step,

$$H = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{4.10}$$

The noise in the measurement is v_k and the state vector is,

$$\hat{x}_k = \begin{bmatrix} \hat{\theta}_k \\ \\ \hat{\omega}_k \end{bmatrix} \tag{4.11}$$

Since the values of Φ_{st} and H are known, the values for Q and R must be chosen and P_0 must be initialized. The process noise covariance matrix, Q, is populated with non-zero diagonal values since there is known modeling errors. Additionally, the position and angular velocity of the motor shaft vary with each other. The covariance values in Q are non-zero,

$$Q = \begin{bmatrix} \sigma_\theta^2 & \sigma_{\theta\omega} \\ \\ \sigma_{\theta\omega} & \sigma_\omega^2 \end{bmatrix} = \begin{bmatrix} 10 & 10 \\ \\ 10 & 10 \end{bmatrix} \tag{4.12}$$

The initial error covariance estimate, P_0, is intialized with the same values as the Q matrix. R is set according to the estimated measurement error. R should account for the quantization noise,

$$R = \sigma_{meas}^2 = 100 \tag{4.13}$$

An offline discrete KF is applied to noisy wheel speed data. Figure 4.3 illustrates an improved wheel speed estimate by the KF described in this section. A velocity profile that grows and decays is used as the set point velocity for the wheel speed controller. The KF wheel speed estimation shows a significant reduction in noise in comparison to the finite-difference estimation method. An online discrete KF implemented on the OpenDani's RedBoard was also found to successfully estimate wheel speed under a variety of inputs. The improved online wheel speed estimate for a sinusoidal velocity input is presented in Figure 4.4. There is, once

again, a significant reduction in noise within the speed estimate.

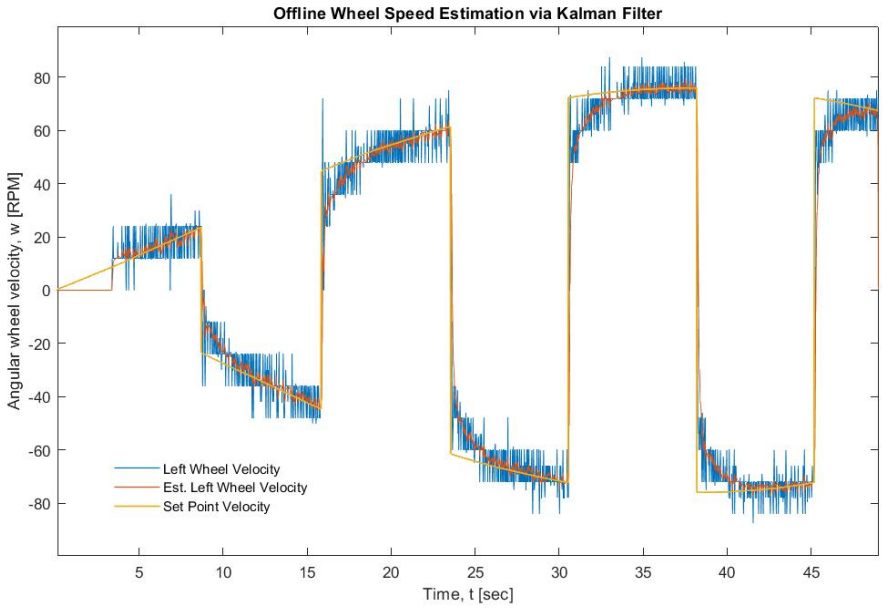

Figure 4.3: Quantization noise from the low-resolution wheel encoders introduces the need for signal processing methods in the OpenDani low-level control. The finite-difference speed estimate is replaced by a KF-based speed estimate to improve odometry.

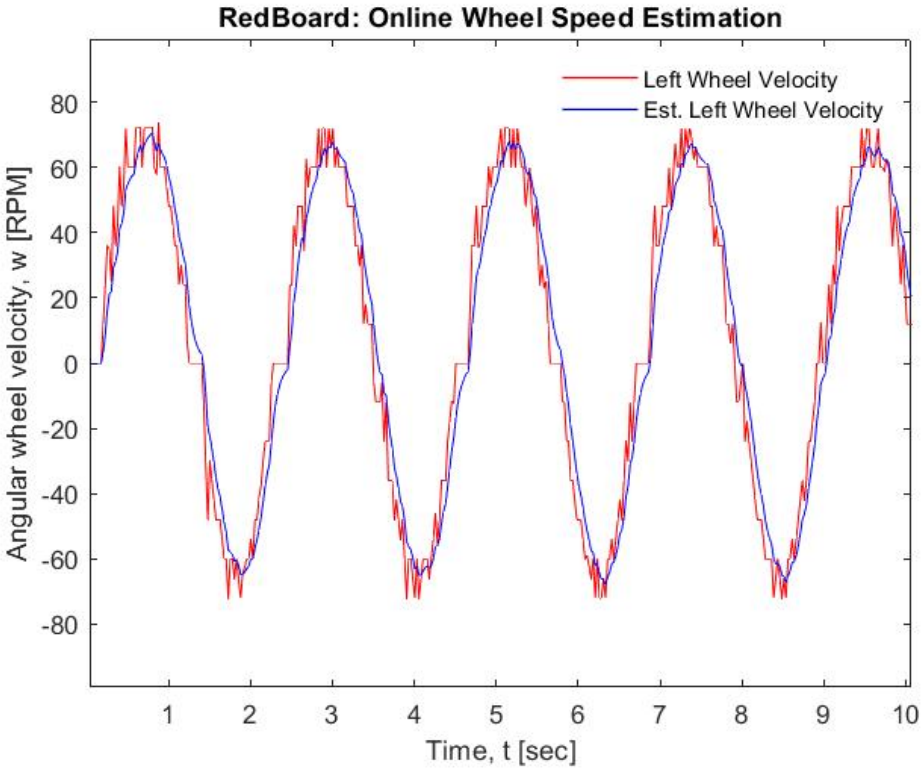

Figure 4.4: A recursive discrete KF formulation is implemented directly on the RedBoard. The filtered speed estimate is used as feedback into the wheel speed controller.

Chapter 5

Obstacle Avoidance

5.1 Introduction

The vDani was developed to execute an *offline* motion planning task [4]. As such, the vDani did not require the ability to make decisions in real-time. To demonstrate this ability in the OpenDani, a LIDAR-based, greedy, local obstacle avoidance (OA) algorithm [44] is deployed on the system. The algorithm is 'greedy' in that a best-score path is chosen from a set of kinematically feasible paths. This is done by employing a simple scoring function. The OA algorithm is considered local as it is limited to the sensing range and field-of-view (FOV) of the LIDAR. Prior knowledge of obstacle locations is not necessary for successful OA execution. Motion of the OpenDani was restricted to forward robot velocities for this study, but the algorithm can be readily extended. The algorithm is implemented through A C++ ROS software node that communicates with the LIDAR and odometry software nodes.

Classic OA methods such as the Dynamic Window Approach [45] utilize a pre-existing map of obstacles to determine the set of admissible velocities for a

mobile robot. Given a map of obstacle locations, the admissible velocities are the set of wheel velocities that grant the robot sufficient space or time to stop before a collision occurs. The lack of perception sensors for the DWA OA method does not allow for corrective behaviors in robot movement. LIDAR-based OA methods enable distance-to-object data to be utilized as feedback into the motion control loop. Another example of a LIDAR-based OA method is based on the Expanded Guide Circle (EGC) method which adopts a selective decision making process to intelligently bypass or enter gaps in an obstacle space [46]. Gap centers are identified using LIDAR scan data.

The greedy OA algorithm used in this work builds a point cloud from LIDAR scans. The point cloud is divided into non-obstacle points and obstacle points. The obstacle-points are further sorted into their possible collision type. The sorting of the point cloud allows for a pre-processing of LIDAR data that is necessary to employ a path scoring function. In the following sections, individual components of the OA algorithm are elaborated on in detail to provide contextual background for the future safety analysis performed on the OpenDani.

5.2 Building a Point Cloud from LIDAR data

To execute the local OA, the LIDAR data must be stored into a point cloud for analysis by the algorithm. The RPLIDAR A1 LIDAR scanner (Shanghai Slamtec Co., Ltd., Shanghai, China) defaults to a 360° FOV at a resolution of 1°. A single scan returns an array of size 360 with radial distance values, d, being within the sensing range of the LIDAR. The sensing range for RPLIDAR A1 is between 0.15 m to 12 m with a resolution of about 0.5 mm. The FOV is the angular range measured with respect to the x-axis of the LIDAR frame shown in Figure 5.1.

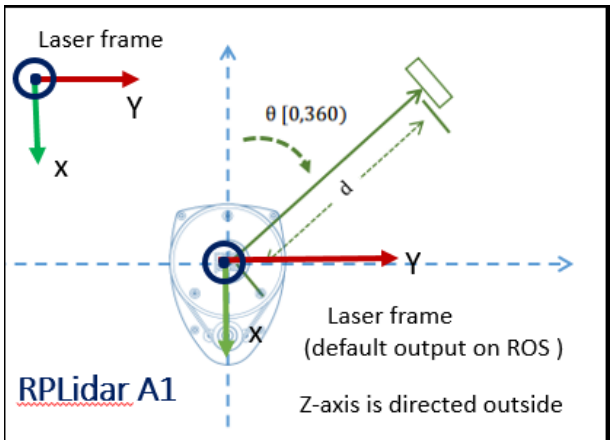

Figure 5.1: The RPLIDAR A1 coordinate frame is useful for building a point cloud from angle and range data. Source: [5]

The angular range as given by the scan topic published by the ROS Lidar node [5] is $[-\pi, \pi]$. To build a point cloud from the LIDAR scan, we project the scan points into the LIDAR frame using the following equation:

$$\begin{bmatrix} x_{scan} \\ y_{scan} \end{bmatrix} = \begin{bmatrix} d\cos\theta_d \\ d\sin\theta_d \end{bmatrix} \tag{5.1}$$

Then, we perform two transformations on the point cloud data to project the points into the OpenDani's baselink frame. The baselink frame origin is at the center of the robot's wheel axle, thus,

$$\begin{bmatrix} x_{obstacle} \\ y_{obstacle} \end{bmatrix} = \begin{bmatrix} \cos\pi & -\sin\pi \\ \sin\pi & \cos\pi \end{bmatrix} \begin{bmatrix} x_{scan} \\ y_{scan} \end{bmatrix} + \begin{bmatrix} x_{laser} \\ y_{laser} \end{bmatrix} \tag{5.2}$$

Due to limited mounting surface area on the OpenDani, the asymmetrical LIDAR sensor was mounted backwards. First, a rotation transform is applied to the point cloud to correct the misalignment of the x-axis in the LIDAR frame with the x-

axis of the baselink frame. A translation is applied to the scan data to accommodate the LIDAR's offset placement from the origin of the baselink frame. Because we only execute OA when traversing forwards, all point cloud data that do not lie in front of the OpenDani's wheel axle can be removed from the cloud.

5.3 Sorting the Point Cloud to Identify Obstacles

Once the initial point cloud is obtained, the points in the cloud are sorted according to whether they are a potential obstacle or not. The criteria for a point to be considered an obstacle is that it must lie in front of the vehicle or within the inner and outer *collision circles* of the vehicle. The inner and outer collision circles are traced by the innermost and outermost dimensions of the vehicle upon executing a path along a turning radius, R. The dimensions of the vehicle are inflated for safety. These inflated dimensions can be referred to as the OpenDani's safety bumper. An analysis of the dimensions that should be chosen for a safe and reasonable safety bumper will be presented in Chapter 6. Figure 5.2 depicts the collision circles that are used to determine whether a point is a potential obstacle.

After a point is determined to be a potential obstacle it is further sorted by the type of obstacle. An obstacle is either of a *front* type or *side* type. The type of obstacle is once again dependent on whether it lies between a series of *sorting* circles traced by the vehicle dimensions. These obstacle sorting circles are illustrated in Figure 5.3. For the straight driving case only front obstacles exist. When driving straight, if a point lies in the area outlined by a box with chosen dimensions it is considered a front obstacle.

The turning radius, R, is measured along the baselink's y-axis and is a crucial parameter in the OA algorithm. It is also referred to as the ICC, and is used to

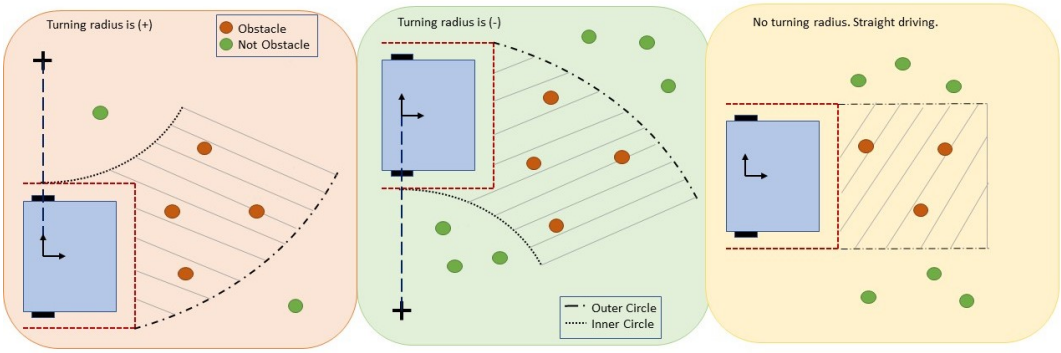

Figure 5.2: The inflated vehicle dimensions are crucial in analyzing the LIDAR point cloud for possible collision hazards to the vehicle. The collision circles are uniquely traced by the vehicle dimensions for each path about the ICC.

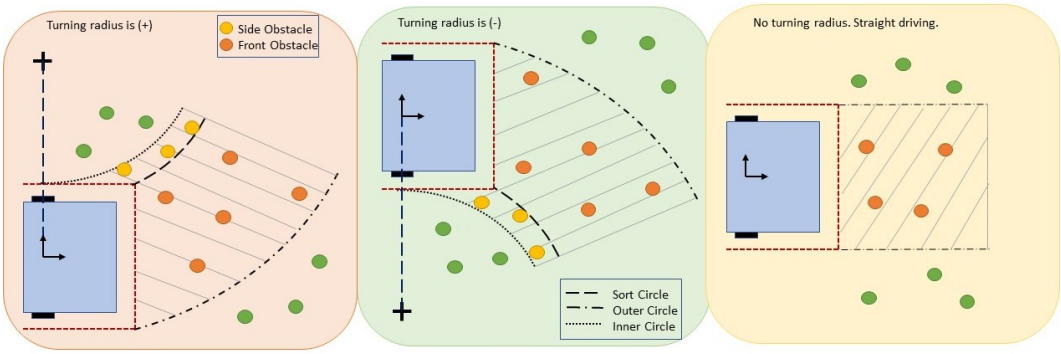

Figure 5.3: A sorting circle is traced by the inner, front corner of the vehicle as it turns about the ICC. Side and front obstacles are distinguished using the sorting circle that divides the area between the two collision circles.

determine whether an object lies within the bounds of the collision circle and the sorting regions. The Euclidean distance between the ICC and the innermost and outermost collision points are used to determine the radii of the tracing circles.

5.4 Kinematically Feasible Paths

Now that the point cloud data is processed and sorted, the OpenDani is given a finite number of paths to evaluate for obstacle avoidance. The paths must not violate the R_{min} constraints of the OpenDani discussed in Chapter 2. For each path, there is a set of inputs to the robot $[v, \frac{v}{R}]$. The turning radius of the vehicle, R, and the curvature of a given path, C, have an inverse relationship,

$$C = \frac{1}{R}. \tag{5.3}$$

A negative curvature indicates a right turning path while a positive curvature indicates a left turning path. When $C = 0$, the robot moves in a straight line and without angular velocity. The paths begin at the baselink of the OpenDani as shown in Figure 5.4. The virtual length of the paths must be chosen carefully. In Chapter 6, the minimum LIDAR range of a differential drive robot is discussed in detail. This design requirement informs the virtual lengths of the paths. A length of 2 m was chosen as the default path length when the vehicle has a LIDAR range outlook of up to 4 m. The path endpoints lie within the LIDAR FOV given these specifications.

The number of kinematically feasible paths that are considered in OA algorithm can be varied. If there are too few paths, the robot is restricted to a very small control input space. The robot's motion may lack continuity with a low number of path options. Given too many path options the OA algorithm may become

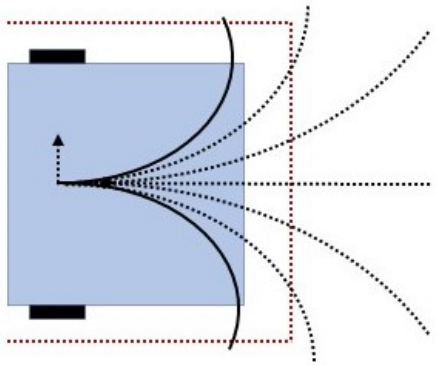

Figure 5.4: All feasible paths must respect the R_{min} constraint of the vehicle. The paths begin at the baselink origin and end at a defined length along each path, l_{path}.

computationally expensive. The scoring function may give multiple paths the same score when there are too many paths being scored. This is especially true if the data type of the variable containing path scores is of low precision.

5.5 Greedily Choosing a Path

Given a set of paths to evaluate, the greedy OA algorithm uses a scoring function to choose the best path in the set. The scoring function is the weighted sum of three factors. The first factor is the length of a path up to the point it intersects an obstacle. This length is referred to as the *free path length* (fpl). The second factor is the *clearance* of the vehicle along a given path. The final factor, the *distance to goal*, considers the OpenDani's overall objective of moving towards a goal. The scoring function is formulated by applying an importance weight to the factors:

$$\text{score} = \text{fpl} + w_1 \times \text{clearance} + w_2 \times (\text{distance to goal}) \tag{5.4}$$

5.5.1 Free Path Length

The *free path length* (fpl) is calculated for each path in the set. For both obstacle types, the process of finding the fpl begins with the angle, θ_{obs}, highlighted in Figure 5.5. This is the angle between the observed obstacle point, the ICC, and the baselink origin. The distance between the ICC and the baselink origin is the

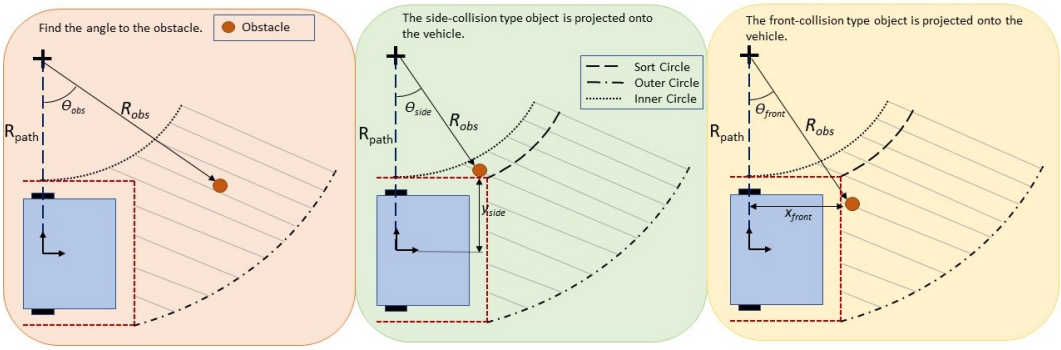

Figure 5.5: For each path in the set, the obstacles are projected onto the vehicle. This is analogous to assuming the vehicle will continue along the path and collide into the obstacles. The projection of the obstacles is useful for finding fpl.

turning radius and is referred to as R_{path} in Figure 5.5. Given that R_{path} is the R of OpenDani along a path, the following equation allows us to find θ_{obs}:

$$\theta_{obs} = \text{atan2}(\frac{x_{obstacle}}{R_{path} - y_{obstacle}}) \tag{5.5}$$

Then, depending on the obstacle type, the obstacle is projected onto the side or front of the vehicle. The projected location of the obstacle is used to determine the obstacle angle at the point of collision, θ_{side} or θ_{front}. These angles are also highlighted in Figure 5.5 as a visual reference.

For a side collision, the y-coordinate of the projected obstacle point, y_{side}, is fixed and always known. The magnitude of y_{side} is half of the inflated vehicle width.

Conveniently, the Euclidean distance between an obstacle point and the vehicle's ICC is constant along a curved path from the set. This distance, R_{obs}, is provided to a geometric relation to find the θ_{side} for a side obstacle:

$$\theta_{side} = \cos^{-1}(\frac{R_{path} - y_{side}}{R_{obs}}) \tag{5.6}$$

For a front obstacle, the x-coordinate of the projected obstacle point, x_{front}, is also fixed and known along a path. The magnitude of x_{front} is the distance from the baselink origin to the inflated front bumper of the vehicle. Again, we use R_{obs} and x_{front} to find the angle to the front obstacle at the point of collision, θ_{front}:

$$\theta_{front} = \sin^{-1}(\frac{x_{front}}{R_{obs}}) \tag{5.7}$$

Finally, θ_{fpl} is defined as the angular difference between $\theta_{obstacle}$ and θ_{side} or θ_{front}.

$$\begin{gathered} \theta_{fpl} = \theta_{obs} - \theta_{side} \\ \text{or} \\ \theta_{fpl} = \theta_{obs} - \theta_{front} \end{gathered} \tag{5.8}$$

The arc length formula is applied to calculate the fpl along each path in the set with curvature, c:

$$\text{fpl} = R \times \theta_{fpl} = \frac{1}{c} \times \theta_{fpl} \tag{5.9}$$

When analyzing the straight path case, the fpl is simply taken as the distance between the obstacle point and the inflated front bumper. Since fpl is evaluated for every point in the cloud and for every path in the set, only the minimum fpl value found is assigned to each respective path.

5.5.2 Clearance

Determining the *clearance* of a given path is straightforward. The clearance assigned to a path is the *minimum* distance between the collision circles and any scan data deemed a non-obstacle. The outer collision circle has a radius R_{outer}. The inner collision circle has a radius R_{inner}. The distance from the ICC to a non-obstacle point, R_{env}, is an important metric for calculating clearance. A representation of clearance for a curved path is given in Figure 5.6. The clearance distance for a single non-obstacle point is calculated with the following conditions:

$$clearance = \begin{cases} R_{env} - R_{outer}, & \text{if } R_{env} \geq R_{outer} \\ R_{inner} - R_{env}, & \text{if } R_{env} \leq R_{inner} \end{cases} \tag{5.10}$$

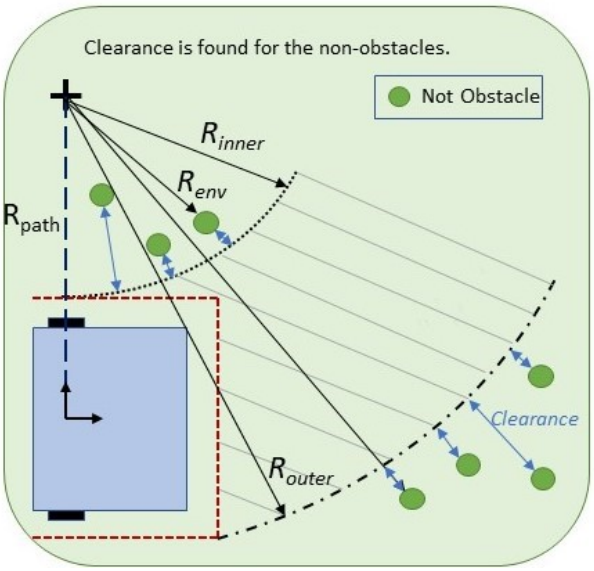

Figure 5.6: The clearance to non-obstacles is calculated for each point in the point cloud. Only the minimum clearance value along each path is stored for use in the scoring function.

Similar to fpl calculation, the straight path clearance is calculated separately. This case is governed by the safety bumper dimensions of the vehicle. Recalling that the width of OpenDani, ω, is inflated by some value of the safety bumper, b, the straight path clearance for a non-obstacle point is once again analyzed in the baselink frame of the vehicle. For a non-obstacle with the following co-ordinates, (x_{env},y_{env}), the straight path clearance is given as such:

$$clearance = \begin{cases} y_{env} - (\frac{w}{2} + b), & \text{if } y_{env} \geq \frac{w}{2} + b \\ |y_{env} - (-\frac{w}{2} - b)|, & \text{if } y_{env} \leq -\frac{w}{2} - b \end{cases} \tag{5.11}$$

5.5.3 Distance to Goal

The *distance to goal* (DTG) factor is especially useful when the local OA is integrated with a global path planner. The DTG factor enables the OA algorithm to consider the robot's proximity to a way point. In this way, the OA algorithm functions as a simplified version of the pure pursuit algorithm mentioned in Chapter 3. When the DTG weight in the scoring function, $w2$, is significantly larger than the clearance weight, $w1$, the OA algorithm prioritizes reaching a goal over traversing the safest path. In Figure 5.7, a star represents the goal in the baselink frame.

Because all possible paths begin at the baselink origin, the end point of each path is easily found within the baselink frame. The path end point serves as an important parameter for the DTG calculation. By restricting the length for each path to some value, l_{path}, the angle formed by the baselink origin, the ICC, and the end of each path is:

$$\theta_{pathend} = \frac{l_{path}}{R_{path}} \tag{5.12}$$

This angle is used to determine the coordinates of each path's endpoint, ($x_{pathend}$,$y_{pathend}$):

$$\begin{aligned} x_{pathend} &= |R_{path}| \times \sin(\theta_{pathend}) \\ y_{pathend} &= |R_{path}| \times (1 - \cos(\theta_{pathend})) \end{aligned} \tag{5.13}$$

Given a set of waypoints in a global map frame, the way points must be transformed into the baselink frame. As the vehicle traverses along the series of waypoints, the waypoint closest to the robot is assigned as the goal point. The Euclidean distance between path ends and the goal point is the DTG. For implementation of the local OA without a global plan, a floating goal point is placed at the coordinates $(4, 0)$ in the OpenDani's baselink frame. This induces a *chasing the carrot* behavior on the system. Therefore, the vehicle primarily drives according to clearance and fpl values with a preference for forward movement.

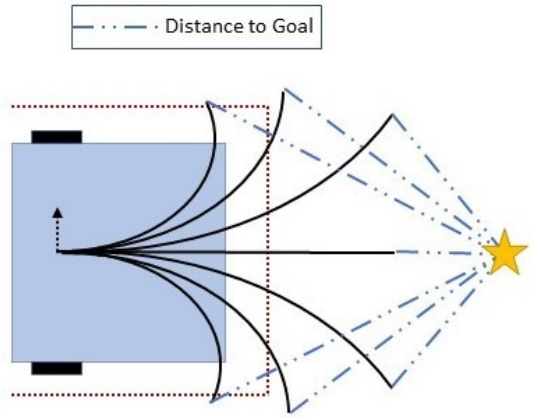

Figure 5.7: The distance between the endpoint of each path and the goal in the baselink frame is used as the distance to goal metric in the scoring function.

5.6 Path Execution via 1D Time Optimal Control

Once a best score path is chosen, the OpenDani must execute the path using 1-dimensional (1D) time optimal control (TOC). The method presented is time optimal in that maximum acceleration (a_{max}), deceleration (d_{max}), and linear velocity (v_{max}) are executed along a path when possible. The stopping length, l_{stop}, of the OpenDani is a crucial safety metric during OA operations,

$$l_{stop} = \frac{v^2}{2d_{max}}, \tag{5.14}$$

where l_{stop} is used to determine the forward velocity, v, of the OpenDani at each time step of the motion control loop, t_{cl}. To implement 1D TOC, the fpl of the best scored path is used as feedback into the system. For the current path, l_{stop} governs the next control input, v, by comparing it to the fpl of the path. The conditional statements below summarize the process of choosing v:

$$v = \begin{cases} v_{prior} + a \times t_{cl}, & \text{if fpl} > l_{stop} \text{ and } v < v_{max} \\ v_{max}, & \text{if fpl} > l_{stop} \text{ and } v \geq v_{max} \\ v_{prior} - d \times t_{cl}, & \text{if fpl} \leq l_{stop} \\ 0, & \text{if fpl} = 0 \end{cases} \tag{5.15}$$

In summary, the OpenDani attempts to execute a trapezoidal velocity profile for v along a path. The values of a_{max} and d_{max} are limited by the actuators. The final a_{max} and d_{max} values do not have to be set to the actuator limits and instead can be set as parameters for the OA algorithm. An example of the velocity profiles that can be executed by the OpenDani are presented in Figure 5.8.

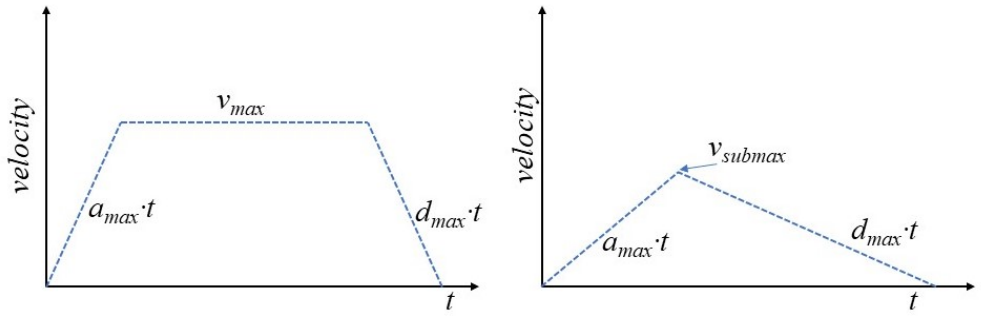

Figure 5.8: There are two expected velocity profiles of the OpenDani while driving under the 1D TOC implementation.

The v_{max} of the OpenDani is determined by the net torques experienced at each wheel as well as the effective load on the vehicle. The net torques are resultants of the reaction forces induced by the OpenDani's mass. A simple test on a chassis dynamometer shows the forward speed of the OpenDani for the rated voltage input, 12 V, to be 0.69 m/s. Note that the the top speed found in this test does not account for inertial effects experienced when traversing on ground. The setup of the chassis dynamometer test can be seen in Figure 5.9.

Because the OpenDani relies on a difference in wheel speeds to turn, the v_{max} of the vehicle is not just limited by the top speed provided by dynamometer test. Meaning, the v_{max} of 0.69 m/s cannot be executed along all paths. The reliable angular speed range of each wheel is between w_{min} and w_{max}, or 80 rpm and 130 rpm, respectively. When driving one of the OpenDani wheels at w_{min} and the other at w_{max}, the minimum turning radius, R_{min}, is 0.71 m. The OpenDani's true v_{max} is 0.56 m/s as restricted by the R_{min} constraint. The odometry report velocity profile of the OpenDani during a simple collision avoidance, stop maneuver is presented in Figure 5.10. The velocity profile matches the expected trapezoidal velocity profile from 1D-TOC.

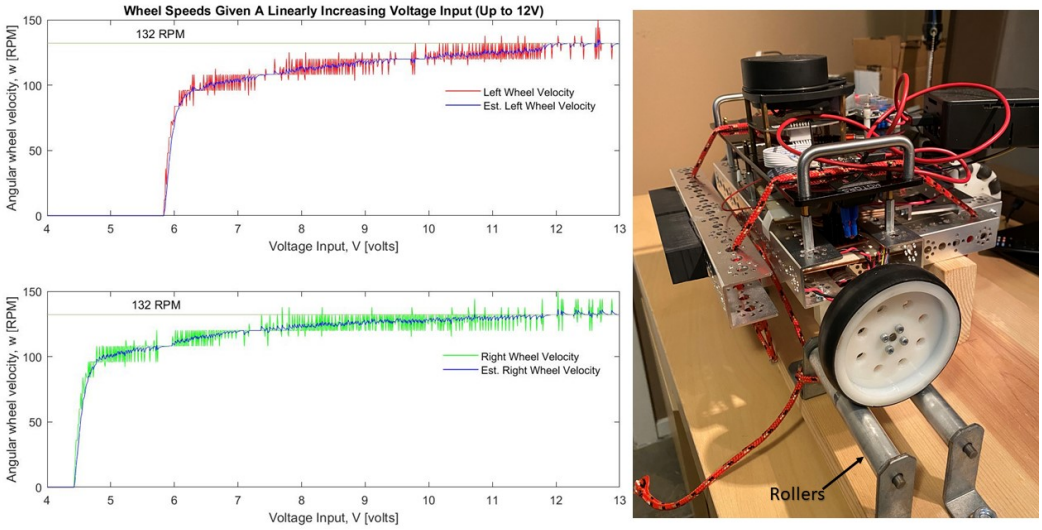

Figure 5.9: The OpenDani is placed on a chassis dynamometer. An open loop, voltage input test is performed to find the top speed of the vehicle.

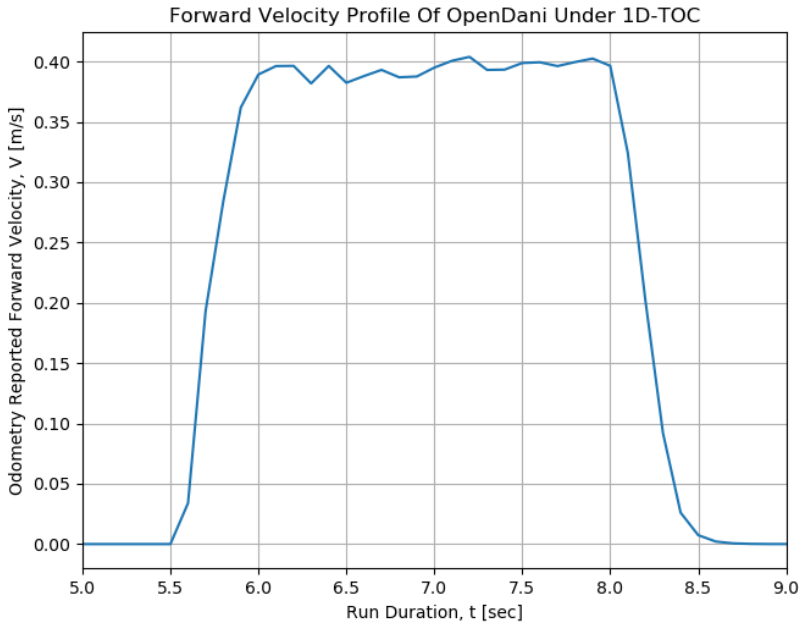

Figure 5.10: During a 1m test run, the OpenDani executes a trapezoidal velocity profile with a max velocity of 0.4 m/s. These results are a product of the logic used in Equation 5.15.

Chapter 6

Safety Experiments

6.1 Introduction

The OpenDani can avoid unmapped, static obstacles under a LIDAR-based obstacle avoidance (OA) algorithm. A safety analysis of the OpenDani within the scope of OA is conducted through the inspection of the system's sensor suite, motion algorithm, and overall wheeled robot design. In inspecting these three factors, insight into the minimum sensor and algorithm requirements for similar MWRs is provided. It is not sufficient to simply demonstrate successful OA execution. Concentrated experiments will be conducted to highlight safety concerns as they relate to uncertainties stemming from the OpenDani design. In mobile robotics, key uncertainties causing safety concerns to the robot hardware include motion uncertainty, environmental uncertainty, and perception uncertainty. As stated in the objectives of this work, safety concerns arising from humans is outside the scope for this study.

Motion uncertainty can be defined as a deviation in the robot drive motion from that which was instructed or expected. For a WMR, a common source of motion uncertainty is the wheel slip phenomenon. In the work [6], a WMR's wheel

speed controller error was assumed to be the dominant source of motion uncertainty. A similar approach is adopted in this work to define the safety bumper utilized for the OA algorithm from Chapter 5. Actuation latency is observed to be an additional source of motion uncertainty in the OpenDani system. We aim to predict this source of motion uncertainty away as done in [47] through a forward prediction method.

Environmental uncertainty can be described by the unexpected presence or movement of objects within the WMRs workspace. It is common to have a pre-mapped environment for WMR navigation. Environmental uncertainty in map-based methods, such as path planning, increases as the real environment deviates from the pre-mapped environment. Because the OpenDani is studied under OA of *static* obstacles only, environmental uncertainty from the unknown velocities and trajectories of dynamic obstacles is non-existent. Only the presence of static obstacles in the undiscovered robot workspace contributes to the environmental uncertainty. Noticeably, the OpenDani does not have an environment map for OA. However, the OpenDani discovers its environment through an on-board LIDAR sensor. As it turns out, for safe OA, only a portion of the environment surrounding the OpenDani must be captured by the LIDAR sensor to reduce environmental uncertainty to a manageable amount. To minimize environmental uncertainty for successful collision avoidance, the LIDAR range must be set accordingly so that OpenDani realizes the environment within a critical distance to static obstacles. Environmental conditions for which the OpenDani cannot successfully execute OA are briefly discussed.

Finally, perception uncertainty is determined by the accuracy of a robot's perception sensors or perception-based algorithms. A LIDAR sensor provides direct distance measurements to surrounding objects. The perception uncertainty is avail-

able through the LIDAR sensor noise specifications for the OpenDani. However, camera-based WMR perception requires an in-depth analysis of the imaging sensor to determine perception uncertainty. Take for example the study performed in [48] that evaluated 10 separate RGBD sensors according to a set of metrics that consider robot perception applications.

WMR safety has been researched through a variety of approaches. The risk of potential collisions between a WMR and hidden dynamic obstacles is studied in [49]. Speed constraints are placed on the vehicle according to the derived risk of potential collisions. A *time-to-collision* metric is formulated in [50] to quantify the severity of near traffic incidents.

The safety analysis presented in the following sections is structured in the following manner. First, possible sources of motion and environmental uncertainties are inspected individually. Methods to mitigate these uncertainties are presented, applied, and verified through isolated experiments. In the isolated experiments, four trials are run to express repeatability of results for each test type. The isolated experiments serve to highlight collisions under the lack of consideration for motion and environmental uncertainties. Only then is the OpenDani tested under the full context of obstacle avoidance. Perception uncertainty is briefly discussed for the OpenDani.

6.2 Evaluating System Actuation Latency

Latency, more specifically actuation latency, can contribute to undesired movement in a robotic system. Actuation latency is the time delay between the instant a control command is sent to a system and the instant that same control command is executed. The undesired movement caused by actuation latency is often realized in

the form of positional overshoot with respect to a target position. This overshoot is a result of the accumulation of past control commands that have been sent within a latency period, $t_{latency}$. These control commands have yet to be executed when the system takes in feedback. In other words, the system does not account for any control commands in the queue when producing future commands. All commands will be executed for the close loop control time period, t_{cl}.

Actuation latency was considered to be a significant factor in developing an accurate, robot pose estimator for the famous RoboCup competition. The *FU-Fighters* team developed a neural-network model to forward predict the vision-based pose estimate for a determined latency period [47]. Overshoot behavior caused by actuation latency is especially dangerous during OA maneuvers since the vehicle may execute longer stopping distances than expected. If $t_{latency}$ is significant or if the vehicle is operating at high speeds, the overshoot behavior is further accentuated. By measuring the actuation latency ($t_{latency}$) of the OpenDani system experimentally, the detrimental overshoot behavior can be avoided.

To determine actuation latency, a sinusoidal forward velocity command signal is sent to the OpenDani. The OpenDani wheels roll freely on a chassis dynamometer in response to velocity commands. The velocity command signal, v_{cmd}, and forward velocity provided by odometry, v_{odom}, are recorded during the experiment. Results from the open-loop, sinusoidal input test can be seen in Figure 6.1. The phase shift between the two signals can be taken as the best estimate of $t_{latency}$. Typically, the difference in the zero crossings of the velocity signals can be used to find the average $t_{latency}$. Because of stiction present at low-speeds in the low-cost motors, it is difficult to find an accurate zero crossing. Instead, we find the $t_{latency}$ by evaluating a *level crossing* at about 0.2 m/s. The average $t_{latency}$ across a set of

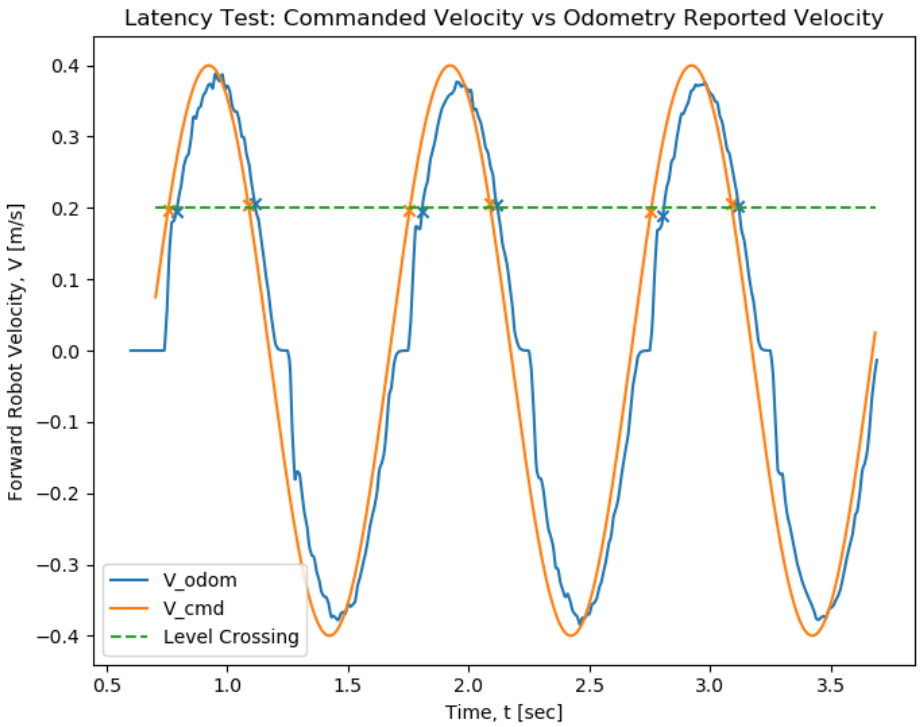

Figure 6.1: The level crossing at which $t_{latency}$ is evaluated is chosen as half the amplitude of the OpenDani velocity command signal.

experiments was found to be about 0.38 seconds. Using this latency period, latency compensation is implemented on the OpenDani.

6.2.1 Latency Compensation

Latency compensation or prediction involves storing all control commands sent to the system within the latency period. The stored commands are used to forward predict the state of the system at $t + t_{latency}$. The forward predicted state is then fed back into the control loop to generate appropriate control commands that consider actuation latency. In the OpenDani's case, the distance traveled along a best chosen path is forward predicted via the stored v_{odom} commands. To evaluate the performance of the OpenDani before and after latency compensation, a simple 1 m drive experiment is performed.

6.2.2 Latency Compensation Drive Test Experiment

A one meter drive experiment was deemed sufficient to assess the drive latency for the OpenDani. The OpenDani is able to reach it's top operational speed for this experiment within this drive distance. In this test, the OpenDani is commanded to drive forward for one meter. Odometry and an external AprilTag vision systems [24] are used to measure the distance traversed by the OpenDani. The AprilTag vision system poses advantages to odometry data as it is isolated from wheel slip. Results indicate that the OpenDani noticeably overshoots the 1 m mark when latency compensation is absent in the motion algorithm. In employing latency compensation, the OpenDani does not overshoot the 1 m mark throughout the experiments. Table 6.1 lists both the AprilTag and odometry reported results for the 1 m drive test under both conditions.

1 Meter OpenDani Drive Test				
	Latency Compensation		No Latency Compensation	
Trial #	Odometry (m)	AprilTag Vision System (m)	Odometry Reading (m)	AprilTag Vision System (m)
1	0.980	0.996	1.093	1.112
2	0.942	0.968	1.089	1.113
3	0.945	0.971	1.128	1.157
4	0.978	0.998	1.133	1.159
Average 1 m Target Error	-0.039	-0.017	0.111	0.135

Table 6.1: An external, computer-vision based measurement system is utilized to compare against the odometry based measurements. The measured state is the distance traversed by the OpenDani vehicle during a 1 m test.

For the trials under latency compensation, there is some slight undershoot in reaching the 1 m target. This is evident through the average error in reaching the target as it is negative for both the odometry and AprilTag reported values. This can be considered a safe behavior compared to an overshooting behavior. A possible source of undershoot for the latency compensated experiments is that, realistically, $t_{latency}$ is not a constant value for the system. Yet, an average value for $t_{latency}$ is used in these experiments. Additionally, wheel slip in the OpenDani can cause some undershoot. The AprilTag measurement system is useful to isolate the effects of wheel slip from the measurement. Wheel slip may also affect experiments without latency compensation. Nevertheless, there is still consistent overshoot in the trials lacking latency compensation. In the worst case, the OpenDani overshoots the 1 m mark by 0.133 m or 0.159 m as reported by odometry and the AprilTag system, respectively.

6.3 Safety Bumper Dimensions Based On Motion Uncertainty

With actuation latency predicted away, the OpenDani design must be further examined to identify sources of motion uncertainty. The OpenDani is currently equipped with an OA functionality. Among OA algorithms, it is common practice to inflate the dimensions of your environment (obstacles, walls) or of the mobile robot itself. This provides a layer of safety to ensure that collisions do not occur when navigating through an obstacle ridden environment. For this work, the external dimensions of the OpenDani are inflated to create a safety bumper. Without the use of a safety bumper, several assumptions are made. The first assumption is that the LIDAR sensor gives perfect, noiseless observations. The second assumption is that the mobile robot perfectly executes the hi-level velocity commands from the OA algorithm. The second assumption includes a "no-slip" assumption.

The assumption that motion uncertainty is dominated by the performance of the wheel velocity controller is leveraged to define the dimensions of the safety bumper. This same assumption forms the basis for OA under motion uncertainty presented in [6]. The controller error of both the left and right wheel velocities are assumed to belong to a Gaussian distribution about the set-point velocity. The work in [6] develops a multi-variate Gaussian distribution model, taking the left and right wheel velocity control errors as the independent variables for the distribution. This joint distribution is then used to inflate the collision areas highlighted within a velocity input space as seen in Figure 6.2. The new velocity input space is used to execute a Dynamic Window Approach (DWA) OA algorithm [45].

The OpenDani does not utilize DWA or a velocity input space, yet it is still useful to examine the low-level wheel velocity controller as a source of motion

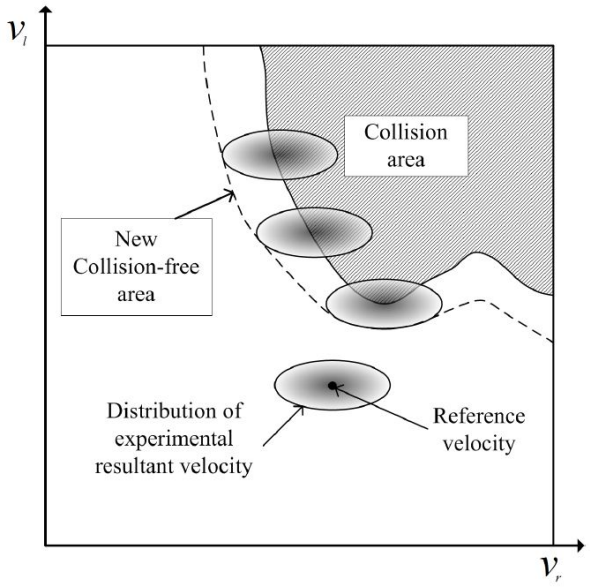

Figure 6.2: The pre-mapped, collision area of a WMR's environment is inflated according to the joint distribution representing wheel speed uncertainties. Figure adapted from [6].

uncertainty.

A critical metric that informs the OA motion algorithm about collision safety is stopping length, l_{stop}, defined as the length traversed by the OpenDani baselink when executing an emergency stop. For a given forward velocity, v,

$$l_{stop} = \frac{v^2}{2d_{max}}, \tag{6.1}$$

and the uncertainty in l_{stop} is propagated using the uncertainty in v. The individual wheel velocity errors obtained by studying the wheel speed controller inform the uncertainty in v. The uncertainty in l_{stop} serves as a guide for setting the dimensions of the safety bumper that surround the left, right, and front sides of the OpenDani.

In order to perform uncertainty propagation, it is assumed that wheel speed error can be described by a Gaussian distribution around the set-point velocity. Wheel velocity errors for either wheel are defined by:

$$vl_{error} = vl_{reference} - vl_{actual} \tag{6.2}$$

$$vr_{error} = vr_{reference} - vr_{actual} \tag{6.3}$$

To examine the wheel speed controller of the OpenDani experimentally, the wheels are driven with a trapezoidal input profile that corresponds to the 1D-TOC velocity profile discussed in Chapter 5. Wheel speed errors are then sampled from this velocity profile. During this experiment, the robot is placed on the chassis dynamometer to assess repeatability and to allows access to the serial port for data logging.

The trapezoidal velocity profiles depicted in Figure 6.3 are defined using a_{max}, d_{max}, and v_{max} parameters used during OA execution and throughout isolated experiments. Table 6.2 lists the values of these parameters as well as the standard

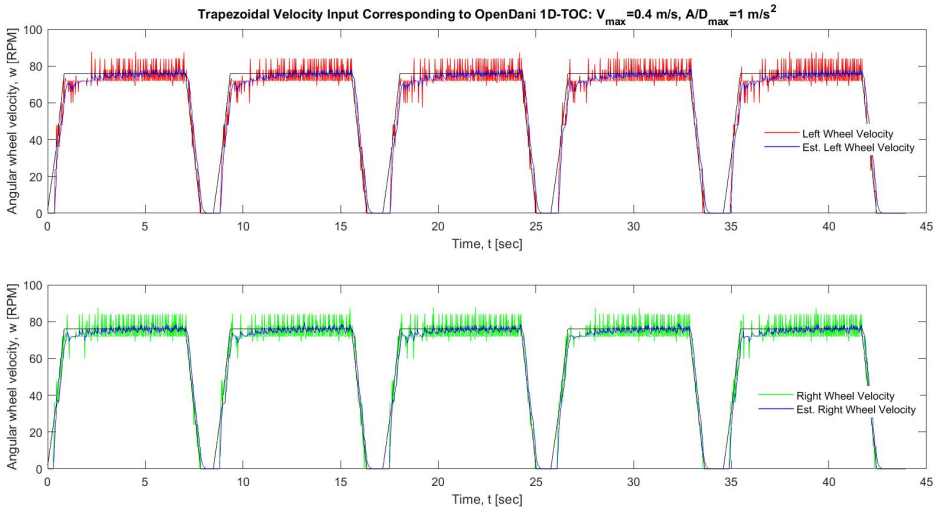

Figure 6.3: The OpenDani's expected velocity profile for 1D-TOC and a given set of robot operating parameters is used to inspect the wheel speed controllers' performance.

deviations of the left and right wheel speed control errors. The PID gains for the left and right wheel speed controllers were individually tuned to achieve comparable dynamic response.

OpenDani Low-Level Controller Experimental Parameters				
v_{max} (m/s)	a_{max} (m/s)	d_{max} (m/s)	Left wheel speed error standard dev., σ_{w_l} (rpm)	Right wheel speed error standard dev., σ_{w_r} (rpm)
0.4	0.4	0.4	6.19	4.79

Table 6.2: It is critical that wheel speed controller performance be examined for each unique set of robot operating parameters. The standard deviations of wheel speed controller errors are expected to vary with these operating parameters.

The standard deviation in the right and left wheel velocities are evaluated from the wheel speed error measurements. For a series of measurements assumed to belong to a Gaussian distribution, the standard uncertainty is the estimated

standard deviation in the measurement. The uncertainty of the OpenDani forward velocity, σ_v, can be calculated from the standard uncertainties of the left and right wheel velocity errors. The combined uncertainty, σ_v, can be taken as,

$$\sigma_v = \frac{\sqrt{\sigma_{v_r}^2 + \sigma_{v_l}^2}}{2}. \tag{6.4}$$

The combined uncertainty in l_{stop} is then propagated from σ_v as such,

$$\sigma_{l_{stop}} = \frac{|2|v_{max}\sigma_v}{2d_{max}}. \tag{6.5}$$

The safety bumper dimensions of the OpenDani can be set according to σl_{stop}, the stopping length uncertainty given any set of OpenDani operational parameters. The combined stopping length uncertainty, σl_{stop}, is found to be about $\pm$ 0.02 m for the set of parameters given in Table 6.2. Instead of directly adopting the σl_{stop} value to inflate the dimensions of the OpenDani, a coverage factor is applied to σl_{stop} to obtain an expanded uncertainty in l_{stop}. The expanded uncertainty value is used to inflate the OpenDani dimensions with a safety bumper length, b,

$$b = 3 \times \sigma_{l_{stop}}. \tag{6.6}$$

A coverage factor of three indicates that the true value of l_{stop} will lie within the lower and upper bound of $l_{stop} \pm 3\sigma_{l_{stop}}$ for 99.73% of measurements taken of l_{stop}. The safety bumper was set to 0.06 m for a series of collision experiments.

6.3.1 Collision Experiments for Evaluating Safety Methods

To examine the effects of latency compensation and the inflation of the OpenDani dimensions via a virtual safety bumper, isolated collision experiments are performed. In the collision experiments, the possible maneuvers taken by OpenDani are a left turn, a right turn, or a straight driving move. An obstacle is placed within the path of the OpenDani for each maneuver type and the OA functionality is tested. The latency compensation and safety bumper settings are varied for each maneuver type as listed in Table 6.3. Collisions are recorded for each each trial.

OpenDani Collision Test: Varying Safety Bumper and Latency Compensation Settings						
O- No Collision **X - Collision**	**Left Turn**		**Straight Drive**		**Right Turn**	
	Latency Comp.	No Latency Comp.	Latency Comp.	No Latency Comp.	Latency Comp.	No Latency Comp.
Safety Bumper: 0.06 m	OOOO	OOOO	OOOO	XXXX	OOOO	OOOO
No Safety Bumper	OOOO	XXXX	OOXX	XXXX	OOOO	XXXX

Table 6.3: An obstacle is placed in the OpenDani's path for a series of maneuvers. Then, the two compensation methods for robot motion uncertainty are added and removed to provide various combinations.

Results from the collision experiments indicate that latency compensation is a minimum requirement for reliable collision avoidance. This conclusion is drawn through the observation that, despite the application of an appropriate safety bumper, the OpenDani collides with the obstacle for all straight driving maneuver trials without latency compensation. The safety bumper alone is not able to compensate for motion uncertainty from the low-level controller and the actuation latency period. Additionally, collisions are observed for all trials and maneuvers when both latency compensation and a safety bumper are absent in the OA algorithm. Though latency

compensation is determined to be a minimum safety requirement in the OpenDani system, it is evident that reliable collision avoidance is not available through latency compensation alone. When examining the straight driving case, multiple collisions are observed while latency compensation is applied without a safety bumper.

The OpenDani is able to avoid collisions solely through the application of an appropriate safety bumper when executing left and right turns. This can be attributed to the larger safety bumper dimensions between the vertex at the front bumper corner of the OpenDani and the vertex at the front bumper corner of the virtual safety bumper. A combination of both an appropriate safety bumper and latency compensation is required for safe OA operations of a WMR. There are no collisions throughout the collision experiments when these two conditions are met.

6.4 Obstacle Avoidance Considering Motion Uncertainty

Two separate sources of motion uncertainty, actuation latency and low-level actuator control error, have been identified. The influence of these factors on the OpenDani's safety have been demonstrated in isolated experiments. It is useful to evaluate these two sources of motion uncertainty under the full context of OA. To do so, the OpenDani is placed in an unmapped, obstacle ridden environment that is pictured in Figure 6.4. The OpenDani uses the greedy OA algorithm described in Chapter 5 to maneuver around the environment. A planned path is not necessary to maneuver the space. The OpenDani's performance during OA is simply evaluated according to whether a collision does or does not occur. The obstacle course is fixed throughout all OA experiments. For the obstacle course experiments, four trials are ran per test condition. We mainly aim to demonstrate that you can effectively set the safety parameters of the motion algorithm to influence performance safety in the

vehicle. In a more extensive study, the number of experiments needed to provide a level of statistical significance would need to be established.

The OA experiment is conducted for the four test conditions:

1) Latency compensation and a safety bumper determined from wheel speed uncertainties are implemented in the motion algorithm.

2) There is no latency compensation in the system but a safety bumper determined by wheel speed uncertainties is provided to the motion algorithm.

3) Latency compensation is implemented in the motion algorithm but there is no safety bumper.

4) Neither latency compensation or a safety bumper are implemented on the OpenDani system.

The OpenDani was able to maneuver through the obstacle ridden environment successfully, without collisions, when it's dimensions were inflated with an appropriate safety bumper and latency compensation was implemented for an experimentally determined $t_{latency}$. The OpenDani safely comes to a stop when it is unable to maneuver around obstacles. The trajectories executed by the OpenDani for the 1st test condition are shown in Figure 6.5.

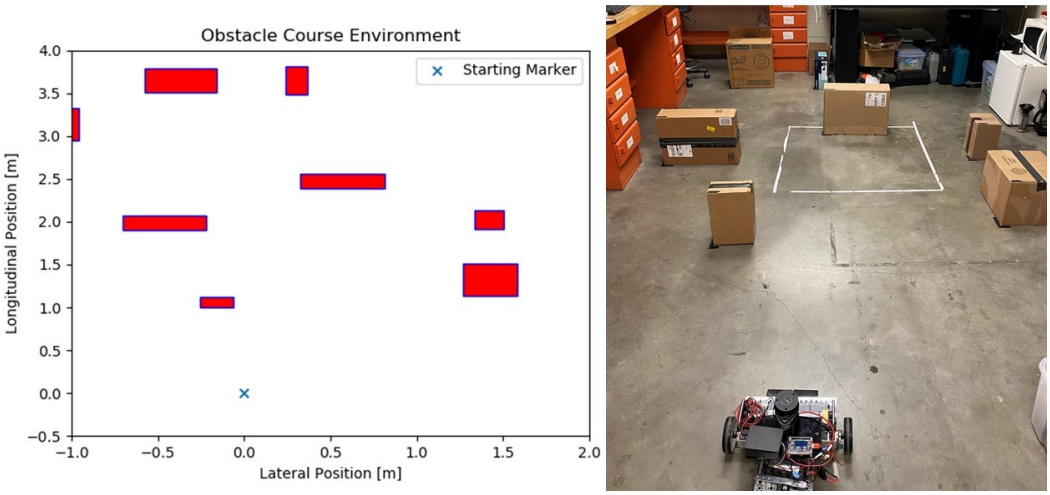

Figure 6.4: Obstacles are placed in the OpenDani's operating environment. The OpenDani does not have prior knowledge of obstacle locations when executing obstacle avoidance.

With latency compensation disabled, the OpenDani collided with obstacles at the end of it's trajectory for three out of four trials. Collisions occur at the front bumper after stop attempts. When comparing trajectories from the fully compensated trials in Figure 6.5 to those for the trials without latency compensation in Figure 6.6, the trajectories in the first case tend to end well before reaching any obstacles. The trajectories in the latter figure clearly enter the gap between the two obstacles at the top left corner of the course. The gap between these obstacles is about 0.423 m which is concerning considering the OpenDani has a width of 0.385 m without a safety bumper. With a safety bumper, the OpenDani width is 0.505 m. Given the inflated dimensions, the OA algorithm realizes there is not enough clearance to continue into the gap but fails to stop before a collision occurs. For one of the trials, the OpenDani is able to successfully avoid all obstacles. Collisions typically occur at the mentioned gap location. Stopping is the only option for safe obstacle avoidance for this area of the course. The successful trial was observed to

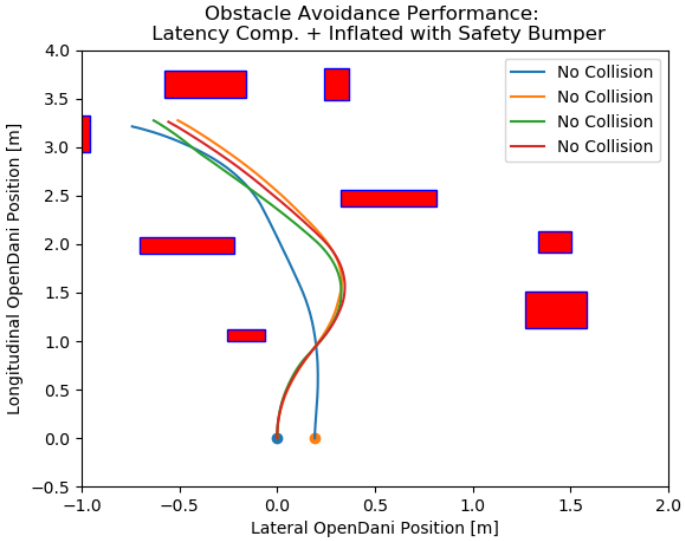

Figure 6.5: The OpenDani is deployed into the obstacle ridden space with latency compensation and an appropriate safety bumper.

be the result of a convenient stopping turn maneuver for which the OpenDani was able to barely miss an obstacle and avoid entering the obstacle gap. This behavior is distinct from the intentional stopping behavior observed in the trials for test condition one. In test condition one, the OpenDani comes to a stop well before entering the gap of concern, in a controlled manner.

OA trials for the third test condition show similar results to those in condition two. The OpenDani collides with obstacles for three out of four trials due to entering a small gap between obstacles. Figure 6.7 documents the trajectories for these trials. The OpenDani does not have inflated dimensions for condition three, which indicates that motion uncertainty from the low-level wheel speed controller is not considered. With an actual vehicle width of 0.423 m, passing through the narrow space at the end of the trajectory is feasible via a perfect approach that aligns the OpenDani center to the center of the gap. The OA algorithm could come to the conclusion

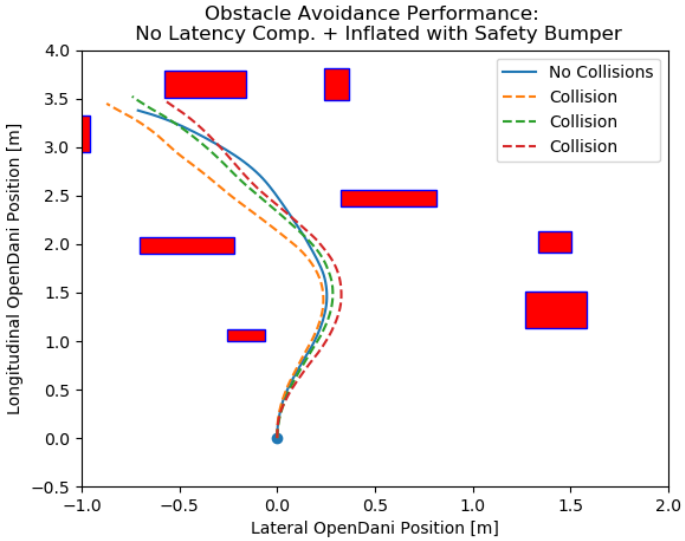

Figure 6.6: Latency compensation is disabled in the OpenDani system. Collision between the OpenDani and obstacles are observed.

that there is sufficient clearance to enter the gap. The absence of a planned path and wheel speed errors contribute to not executing a perfect approach into the gap. In test condition three, collisions typically occur through contact between the OpenDani wheels and an obstacle.

Finally, the OpenDani is tested for test condition four. Without latency compensation or a safety bumper, motion uncertainties are assumed to be non-existent. It is clear from results that this is detrimental to the safe OA operations of the OpenDani. As expected, the OpenDani collides with the obstacle course for all condition four trials. For one instance, the OpenDani collides with a small obstacle at the beginning of it's journey. This obstacle is typically avoided during all other test conditions. The short trajectory from this trial run is visible in Figure 6.8.

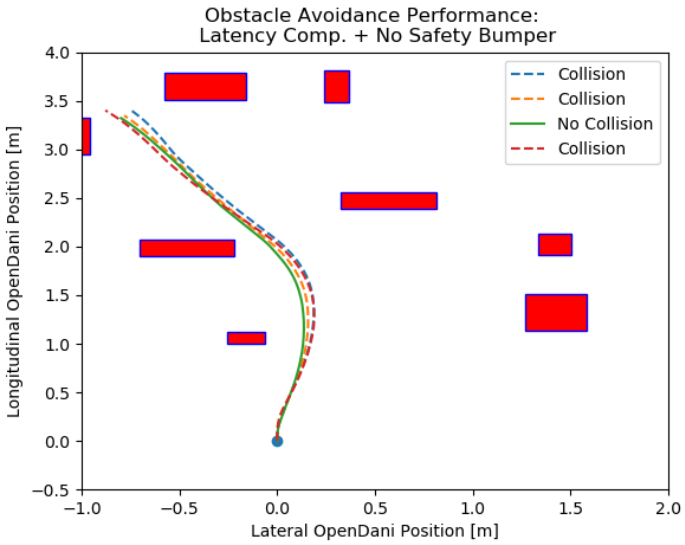

Figure 6.7: The OpenDani dimensions are not inflated to compensate for wheel speed controller error. Latency compensation remains enabled.

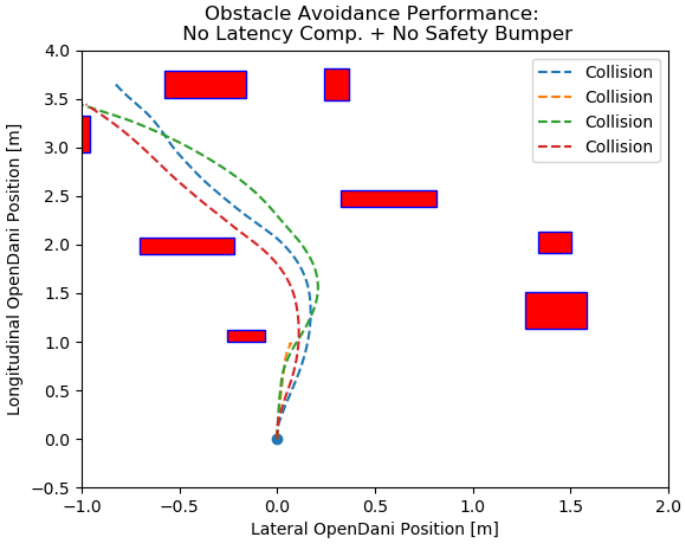

Figure 6.8: The OpenDani is operated without considering sources of motion uncertainty as Latency compensation and a safety bumper are removed from the obstacle avoidance algorithm.

6.5 LIDAR Range Requirement From Robot Design Characteristics

Motion uncertainties influencing the safe operation of the OpenDani have been addressed through latency compensation and the informed inflation of the vehicle dimensions. Additional hazards to WMR safety can be identified in examining the relationship between the OpenDani and it's environmental uncertainty. In the OpenDani, exploration of the surrounding environment is enabled through a 2D LIDAR sensor. Sensing range and sampling rate are critical specifications that guide the appropriate selection of a LIDAR sensor for a WMR. For the OpenDani, an analysis is conducted to determine the minimum LIDAR range requirement ($Range_{min}$) to achieve a safe level of *outlook* for successful OA. The analysis is based on kinematics, the imposed non-holonomic constraints of the vehicle, operating parameters, and robot dimensions. The LIDAR outlook should minimize environmental uncertainty such that static obstacles are avoided. The field of view (FOV) requirement of the OpenDani is briefly discussed. The FOV is the angular range that must be swept by the LIDAR sensor at $Range_{min}$.

It must be highlighted that the OpenDani's LIDAR sensor is mounted on top of it's body, about 0.2 m off the ground. The design choice of mounting the LIDAR on top of the body, as opposed to below or level with the body, restricts the safe operating environment of the OpenDani. The environment should not include low hanging obstacles or short obstacles that cannot be perceived by the LIDAR sensor. Taller objects with a protruding, short base are especially hazardous to OpenDani. Therefore, the environments for which LIDAR-based WMRs can safely operate are restricted by the height of their LIDAR sensor with respect to ground. It is important to consider LIDAR mounting location during the design process of a

WMR. Mounting options include mounting the LIDAR at an angle (not parallel to the ground) so that it may have more vertical coverage. However, a trade-off of this design choice is that the LIDAR range may be shortened. An alternative is to use multiple LIDAR sensors. The $Range_{min}$ analysis is valid for environments where static obstacles are detectable 0.2 m above the ground. The OpenDani must perceive obstacles from a specified distance such that there is sufficient clearance between the vehicle dimensions and the obstacles. This distance can be expressed within the local robot frame, and can be referred to as the *critical view point*. The minimum LIDAR range requirement is set as the Euclidean distance from the LIDAR location, (x_{lidar}, y_{lidar}), to the critical view point, (x_{view}, y_{view}),

$$Range_{min} = \sqrt{(x_{lidar} - x_{view})^2 + (y_{lidar} - y_{view})^2} \tag{6.7}$$

All coordinates are in the baselink frame. The critical view point is found by identifying a suitable clearance distance between the vehicle and obstacles through kinematic analysis.

The OpenDani LIDAR outlook analysis begins with the critical safety metric, l_{stop}, which is the distance traversed by the baselink origin after driving at v_{max} and coming to a complete stop with a maximum deceleration, d_{max}. First, the straight driving collision avoidance case is presented. l_{stop} is set as the necessary clearance distance between the vehicle's front bumper and possible obstacles. $Range_{min}$ would then be the Euclidean distance between (x_{lidar}, y_{lidar}) and the projection of the corner points along the front bumper for a distance of l_{stop}, (x_{bumper}, y_{bumper}),

$$x_{bumper} = l + b + l_{stop} \tag{6.8}$$

$$y_{bumper} = \pm(\frac{w}{2} + b) \tag{6.9}$$

If the OpenDani has yet to reach v_{max}, $Range_{min}$ is a conservative LIDAR range. However, further analysis reveals that the straight driving case is not conservative in determining the critical view point, (x_{view}, y_{view}),

$$x_{view} \neq x_{bumper} \tag{6.10}$$

$$y_{view} \neq y_{bumper} \tag{6.11}$$

To identify the critical view point, the OpenDani must be analyzed under a series of stopping turns with varying turning radii, R. The minimum turning radius, R_{min}, is included as the lower bound of the set of radii. For the OpenDani, it was stated in Chapter 2 that wheel velocities have been restricted to forward velocities. This disables the OpenDani from turning about it's own center or executing a zero-radius turn with $R = 0$. For the OpenDani, R_{min} has a value of 0.7 m. The trajectory of the OpenDani during a turn is demonstrated in Figure 6.9 below.

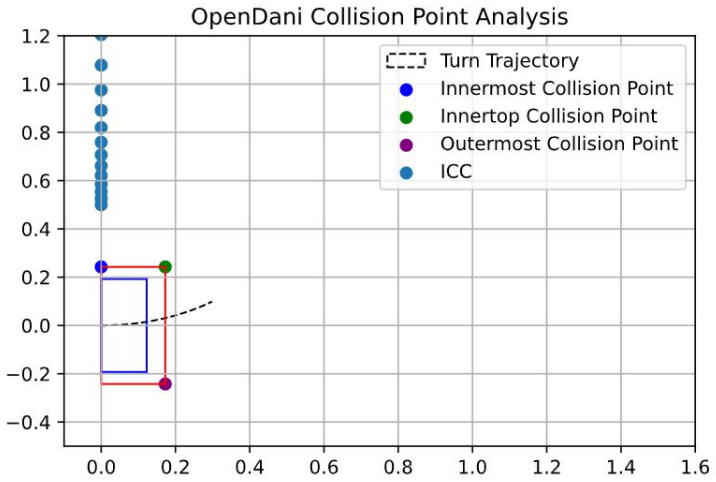

Figure 6.9: The collision points on the OpenDani safety bumper trace circular, virtual paths as the vehicle turns about a ICC. These points are critical for safety analysis.

As in the straight driving case, l_{stop} is useful for determining the necessary clearance distance. However, l_{stop} is now interpreted as the arc length traced by the baselink when executing a stopping turn with radius, R. The l_{stop} safety metric is converted into an angular representation, the stop angle, θ_{stop}. This is the angular displacement of the vehicle about the ICC, $(0, R)$, during a stopping turn, estimated by the arc length formula,

$$\theta_{stop} = \frac{l_{stop}}{R}. \tag{6.12}$$

The specified clearance distance between the OpenDani and potential obstacles remains as l_{stop} but it is measured along the arc lengths traced by the inflated OpenDani dimensions. These tracing points are presented as collision points in Chapter 5 and are highlighted in Figure 6.10. If an object lies in between or intersects these traced arc lengths, a collision is possible. The collision points are projected forward given the angular displacement, θ_{stop}, for a series of turns. Their projections after executing the various stopping turns can be seen in Figure 6.11.

The projection of the outermost collision point surpasses the projection of the safety bumper for the straight driving case for all turns between R_{min} and R_{max}. R_{max} is set as 20 m for this analysis. As R approaches an infinite value, the OpenDani exhibits straight driving behavior. In Figure 6.11, the length traced by the outermost collision point converges towards the obstacle clearance distance for the straight driving case.

The operating conditions and physical dimensions of the OpenDani define the maximum arc length traced by the vehicle during an emergency stopping turn. The outermost collision point will always trace an arc length longer than l_{stop}. The radius of the arc traced by the outer collision point, R_{out}, will always be greater than the turning radius, R. The equation for the maximum traced arc length ($l_{max_{arclength}}$)

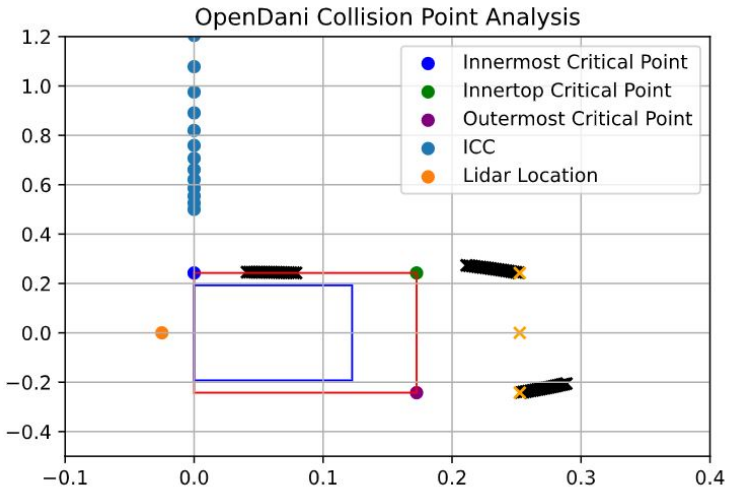

Figure 6.10: The resultant location of the vehicle's critical points, upon executing a critical stopping maneuver, are demonstrated for various turning radii and the straight driving case.

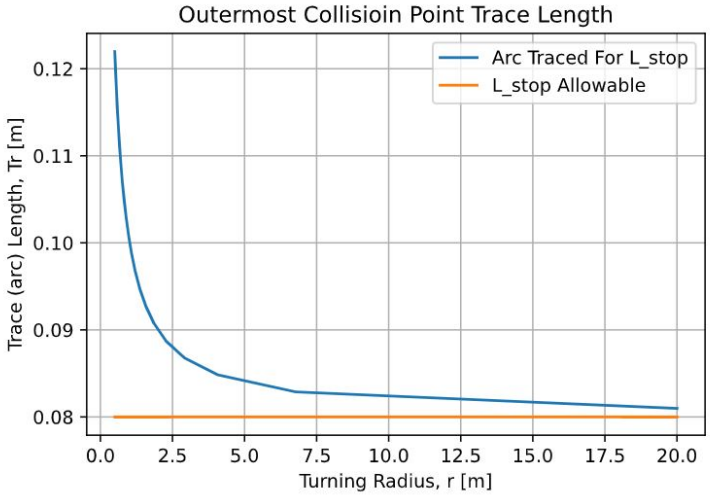

Figure 6.11: The traced path length of the outermost collision point is greatest at the minimum turning radius allowable. This trace length converges towards the straight driving trace length as the vehicle's turning radius increases.

for any stopping turn is,

$$l_{max_{arclength}} = \frac{R_{out}}{R} l_{stop}. \tag{6.13}$$

For the condition where $\theta_{stop} < \frac{\pi}{2}$, the *critical view point* is taken as the end point of the outermost collision point projected along $l_{max_{arclength}}$,

$$x_{view} = R_{out} \sin(\theta_{out} + \theta_{stop}) \tag{6.14}$$

$$y_{view} = R - R_{out} \cos(\theta_{out} + \theta_{stop}) \tag{6.15}$$

where θ_{out} is the angle between the baselink origin, the ICC, and outer collision point on the vehicle. The analysis presented above reveals that for a differential drive vehicle constrained to forward wheel velocities, $Range_{min}$ is established by R_{min}. At R_{min}, θ_{stop} is at its greatest value. As θ_{stop} increases within the range $[0, \frac{\pi}{2}]$, so does $Range_{min}$.

The FOV requirement of the OpenDani is reduced in comparison to a differential drive robot that is able to maneuver backwards. The FOV must always encompass the collision points defined by the OpenDani dimensions. The FOV must span the angular range of $[-\frac{\pi}{2}, \frac{\pi}{2}]$ when the LIDAR is aligned with the wheel axis. As the LIDAR is mounted ahead of the wheel axis, the minimum FOV range increases. As the LIDAR is mounted behind the wheel axis, the minimum FOV range decreases.

6.5.1 A Practical Application of the Minimum LIDAR Range Requirement

The analysis presented above assumes that the OpenDani observes obstacles as soon as they penetrate the LIDAR FOV. This is only true if the LIDAR sensor

is continuously observing. Truly, the OpenDani observes its environment with the LIDAR sampling rate, f_{lidar}. The sampling period, t_{lidar}, is defined by the sampling rate,

$$t_{lidar} = \frac{1}{f_{lidar}} \tag{6.16}$$

Consider the instance where the LIDAR makes an observation such that an object is merely outside the FOV. The OpenDani will continue traveling with some forward velocity, v, between sampling periods. To account for the discrete LIDAR sampling, the distance traversed between sampling periods, d_{obs}, must be augmented to the l_{stop} clearance distance utilized in the analysis above. It is assumed that the OpenDani is at v_{max} when calculating d_{obs},

$$d_{obs} = t_{lidar} \times v_{max}. \tag{6.17}$$

The minimum LIDAR range requirement must consider d_{obs} to account for the discrete sampling scenario. $Range_{min}$ is best found by using the conservative clearance distance, $l_{project}$, to project the collision points for both the straight driving case and the turning case, i.e.,

$$l_{project} = l_{stop} + d_{obs} \tag{6.18}$$

6.6 Verification of Minimum LIDAR Range Requirement

To verify the $Range_{min}$ analysis, collision avoidance experiments were conducted similarly to the experiments conducted for investigating the effects of motion uncertainties. A straight driving collision avoidance test is conducted. The

second collision avoidance test inspects the outer collision corner of the robot when executing a stopping turn at R_{min}. The OpenDani LIDAR range was varied across collision experiments. The proposed, safe minimum LIDAR range requirement, $Range_{min}$, accounts for the clearance distance given a 6.8 Hz LIDAR sampling rate and l_{stop}. The LIDAR range, $Range_{sim}$ assumes continuous observations instead. With continuous LIDAR observations, any possible obstacles are detected at the very instant that they enter the LIDAR's FOV. This is not true in practical situations since LIDAR observations are discrete with a sampling rate, f_{lidar}. For this reason, it is expected that some collisions occur at the $Range_{sim}$ observation range. Finally, $Range_{sub}$ is an unsafe LIDAR range that does not grant the OpenDani sufficient LIDAR lookout for successful obstacle avoidance. Collisions are expected for experiments at $Range_{sub}$. The $Range_{sim}$ and $Range_{min}$ LIDAR range values for each collision test case were found according to the OpenDani operating parameters presented in Table 6.4.

OpenDani OA Operating Parameters for LIDAR Range Analysis				
v_{max} (m/s)	a_{max} (m/s)	d_{max} (m/s)	R_{min} (m)	FOV (°)
0.4	1	1	0.7	±182.08

Table 6.4: The OpenDani operating parameters correspond to the values utilized for LIDAR range analysis and the drive characteristics provided to the vehicle's motion algorithm during obstacle course experiments.

For the $Range_{sub}$ value of the straight driving collision test case, 90% of the $Range_{sim}$ value for the same test case is used. This ensures that the LIDAR range is unsafe for OA. By adopting the $Range_{min}$ value from the straight driving case as the $Range_{sub}$ value of the turning case, the claim that the straight driving case

is not conservative in finding the true value of $Range_{min}$ is verified. Table 6.5 lists the relevant LIDAR ranges used throughout experiments.

OpenDani Experimental LIDAR Range Values		
	Straight Drive	**Turn at R_{min}**
$Range_{min}$	**0.3143 m**	**0.4104 m**
$Range_{sim}$	**0.2550 m**	**0.3618 m**
$Range_{sub}$	**0.2295 m**	**0.3143 m**

Table 6.5: A separate set of LIDAR ranges are examined for the straight driving and minimum turning radius stopping maneuver.

Latency compensation and a suitable safety bumper are applied to the OpenDani as discussed in the beginning of this chapter. An obstacle is placed about 1 m in front of the OpenDani for the straight driving collision avoidance test. The OpenDani is only allowed to avoid objects along a straight path. Results from the experiment are listed in Table 6.6. The OpenDani successfully avoids obstacles for all trials when the LIDAR range is set at $Range_{min}$. Recall that this LIDAR range considers the distance traversed by the vehicle between the LIDAR sampling period. For the straight driving case, the LIDAR range that assumes continuous LIDAR sampling, $Range_{sim}$, collides into obstacles for two of the four of experimental trials. This further highlights the necessity to adapt to the practical case of obtaining discrete LIDAR observations for a WMR.

The setup for collision avoidance experiments during an R_{min} turn places an obstacle along the circular path traced by the outermost collision point of the vehicle. The set up is observed in Figure 6.12, The obstacle is intentionally placed so that it does not lie along the paths traced by the other collision points. This allows the OpenDani to only detect the object as a front type object. Any collisions

OpenDani Minimum LIDAR Range Requirement: Collision Tests		
O- No Collision X - Collision	Straight Drive	Turn at R_{min}
$Range_{min}$	OOOO	OOOO
$Range_{sim}$	OXXO	OOOO
$Range_{sub}$	XXXX	XXXX

Table 6.6: Isolated collision experiments for a variety of OpenDani LIDAR Ranges support the LIDAR range analysis that determines $Range_{min}$.

are expected to occur at the outermost collision point.

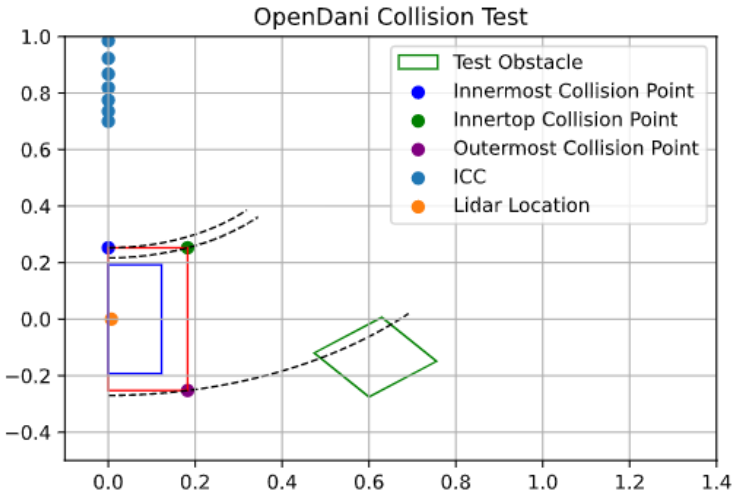

Figure 6.12: For the R_{min} collision experiment, the test obstacle is placed in the path of the outermost collision point but not in the path of the other collision points.

Collisions were not observed for a LIDAR range value of either $Range_{sim}$ or $Range_{min}$ during the R_{min} avoidance turn experiments. Similarly to experimental results from Section 6.3.1, the turning avoidance maneuver seems to encounter less collisions than the straight driving avoidance maneuver. This is likely because the safety bumper at the front corner of the vehicle is inflated at a greater normal

distance with respect to the actual corner of the OpenDani. During the turn experiments, it is observed that the $Range_{min}$ LIDAR range value results in a larger clearance gap between the vehicle and obstacles upon completing the avoidance maneuver. As expected, consistent collisions are seen across all trials for both collision test types when the LIDAR range is set to an unsafe value, $Range_{sub}$. Recall that the $Range_{sub}$ value for the turning avoidance maneuver is equivalent to the $Range_{min}$ value for the straight driving case. For this range, all four turning avoidance trials see the OpenDani collide into the obstacles at the outermost corner of the vehicle. This confirms that the straight driving case is not conservative in finding $Range_{min}$. The R_{min} stopping turn condition sets the final $Range_{min}$ value.

6.7 Obstacle Avoidance with Varying Environmental Uncertainty

The predicted outcomes for collision avoidance performance were verified for a variety of LIDAR range values within isolated collision experiments . The OpenDani must be examined in a realistic OA scenario. The environmental uncertainty of the OpenDani is reduced as the LIDAR range and FOV increases. Since the FOV of the vehicle is fixed, the environmental uncertainty can be varied by adjusting the LIDAR range. LIDAR range and environmental uncertainty have an inverse relationship. A greater LIDAR range reduces environmental uncertainty. To investigate the impact of LIDAR range on successful OA execution, the OpenDani is once again placed into the obstacle course depicted in Figure 6.4. The obstacle course experiments are performed for three distinct LIDAR ranges.

Each LIDAR range used for the obstacle course experiments is a measure of the OpenDani's environmental uncertainty. The first LIDAR range, $Range_{unsafe}$,

represents an environmental uncertainty that is troublesome for the OpenDani. $Range_{unsafe}$ is set to the $Range_{sub}$ value of the straight driving case. The OpenDani is expected to perceive too little of its environment to safely execute OA. Next, a viable amount of environmental uncertainty is chosen by setting the LIDAR range to $Range_{min}$ for the conservative R_{min} turning case. Lastly, the LIDAR range is set to be about double that of $Range_{min}$. This represents a level of environmental uncertainty that is significantly reduced from the safe, baseline amount.

Upon setting the LIDAR range value to $Range_{unsafe}$, collisions with obstacles occur in all four trials. The vehicle pushes through the first obstacle it encounters and continues on its trajectory before colliding with a second obstacle. In one instance, the vehicle collides with a third obstacle. The trajectories of the OpenDani during these experiments are illustrated in Figure 6.13.

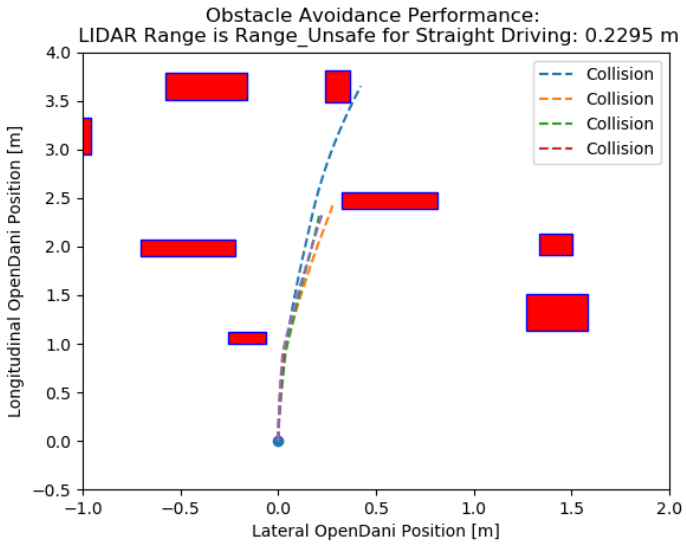

Figure 6.13: The OpenDani collides with the obstacle when it's LIDAR range is set to $Range_{sub}$ for the straight driving case.

While the LIDAR range is set to the suggested minimum requirement value,

$Range_{min}$, the vehicle defaults to a stopping behavior for all trials. The OpenDani has sufficient room to detour around the first obstacle it encounters. Nonetheless, it continues to drive up to the first obstacle before executing an emergency stop. There are no collisions observed throughout these trials. In one instance, a small turning attempt is made before the vehicle comes to a stop. The short trajectories are visible in Figure 6.14. Though the OpenDani has enough LIDAR outlook to avoid collisions with obstacles, the environmental uncertainty is great enough to affect the OA performance of the OpenDani. Recall the scoring function presented in Chapter 5 is used to choose a best path from a set. The degraded performance is a result of the invariability across the *free path length (fpl)* and *clearance* sub-scores determined for each path in the set. With the LIDAR Range value at $Range_{min}$, the *clearance* and *fpl* factors are not likely to vary due to the sparsity of the point cloud used to calculate these factors. The OA algorithm is inclined to choose the straight forward path as a result of the *distance to goal* factor being dominant in the path scoring function. The goal is set as a "hanging carrot" goal directly in front of the vehicle, making the straight forward path the highest scoring path. Since $Range_{min}$ is the shortest possible LIDAR range value, a stop action is induced on the system as soon as obstacles penetrate the LIDAR FOV.

To improve the performance of the OpenDani, the environmental uncertainty is further reduced by setting the LIDAR range to 1 m. This value is more than double the value of $Range_{min}$ determined from the operating parameters of the OpenDani. An improvement in OA performance is exemplified by the longer, collision-free trajectories of the vehicle plotted in Figure 6.15. The OpenDani successfully maneuvers around the obstacle course before safely coming to a stop for all four trials.

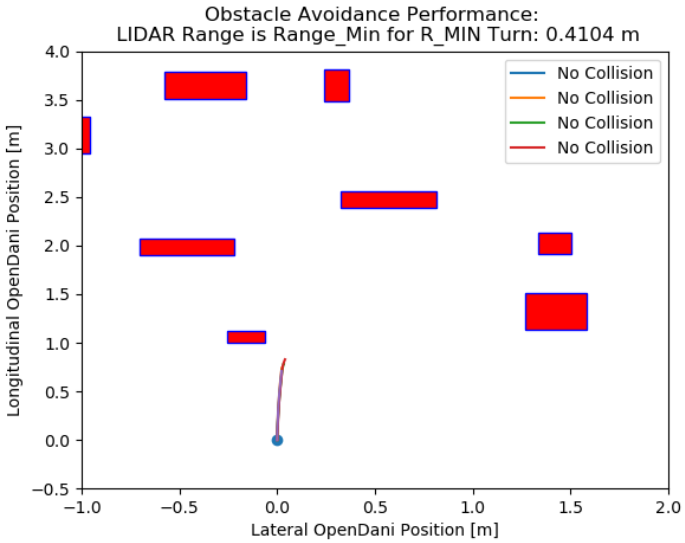

Figure 6.14: Though the OpenDani does not collide with the obstacle course when the LIDAR range is set to $Range_{min}$ for the R_{min} stopping case, the OA performance is degraded.

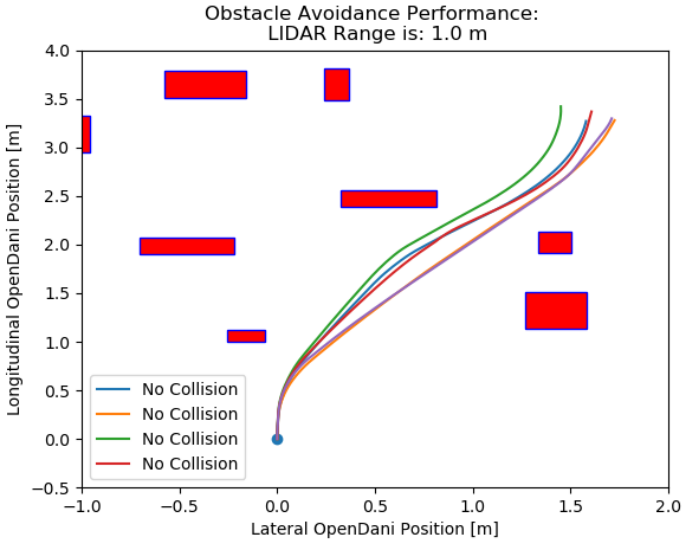

Figure 6.15: The OpenDani is able to maneuver throughout the obstacle course with improved performance when the LIDAR range is set to about double the minimum LIDAR range requirement for safe collision avoidance.

6.7.1 On Perception Uncertainty

The OpenDani's perception uncertainty is characterized by the noise level of the RPLIDAR A1M8 LIDAR sensor. In particular, the range measurement error is of interest. The sensor returns distance values to objects surrounding the robot with a measurement resolution of 0.0005 m. The angular resolution of the sensor is 1°. For this study, the LIDAR sensor's range error was assumed to be compensated for in the coverage factor used in the motion uncertainty analysis.

Chapter 7

Conclusion and Future Work

In this work, a redesign of a WMR was completed to elevate the level of autonomy of a previous design. The new WMR design maintained a low-cost, accessible design through the use of open-source software and plug-and-play sensors. An onboard, Raspberry Pi 4 Linux computer served as a simple sensor interfacing platform that allowed the OpenDani to become an entirely self-contained WMR. A design comparison between the OpenDani and Dani was performed by analyzing each design's ability to support the root autonomous capabilities of an unmanned system as defined by NIST. The OpenDani was placed in an obstacle course environment to demonstrate the increased autonomy. The OpenDani successfully executes online obstacle avoidance within the un-mapped obstacle course whereas the older vDani design was not capable of such a feat.

Furthermore, a safety analysis of the OpenDani under an obstacle avoidance functionality was conducted to verify sources of uncertainty in the WMR that present safety concerns to robot hardware. Methods to compensate for two identified motion uncertainty sources, system actuation latency and low-level wheel speed con-

troller error, were presented. The latency compensation and safety bumper applied to the motion algorithm were found to be necessary for safe and reliable OA. The environmental uncertainty in the OpenDani was found to be related to the LIDAR sensor's range. A kinematic-based analysis of the OpenDani was introduced to determine the minimum LIDAR range requirement for the OpenDani. The minimum LIDAR requirement ($Range_{min}$) is calculated for a unique set of OpenDani operating parameters. The $Range_{min}$ is experimentally validated to be a safe LIDAR range for OA but it is revealed that it is not ideal for OA performance.

In identifying sources of either motion, environmental, or perception uncertainty, the OpenDani safety analysis is approached in a general manner. The hope is that a similar framework can be used to asses the safety of other WMR types. Future work can further asses the effects of sampling rate on the environmental uncertainty within a WMR similar to OpenDani. This can be studied by directly controlling the LIDAR sensor's scan rate during obstacle avoidance. Additionally, motion plans from path planners have a significant influence on robot motion. By integrating a path planner into the OpenDani system, the safety of the planned path can be assessed according to a set of metrics (e.g., the proximity to obstacles). Still, the motion uncertainty in following the planned path is dependent on the low-level wheel speed controller performance.

Bibliography

[1] P. Yazdjerdi and N. Meskin, "Actuator fault detection and isolation of differential drive mobile robots using multiple model algorithm," in *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 0439–0443, IEEE, 2017.

[2] B. H. Tongue and S. Sheppard, *Dynamics: Analysis and design of systems in motion*. John Wiley & Sons, 2005.

[3] "Automated vehicles for safety." `https://www.nhtsa.gov/technology-innovation/automated-vehicles`, Jun 2020.

[4] P. Ainampudi, "Box Pushing with a Mobile Robot Using Visual Servoing," master thesis, The University of Texas at Austin, Main Building (MAI) 110 Inner Campus Drive Austin, TX 78705, May 2019.

[5] Robopeak, "rplidar_ros." `https://github.com/robopeak/rplidar_ros`, Aug. 2019.

[6] J. Jin and W. Chung, "Obstacle avoidance of two-wheel differential robots considering the uncertainty of robot motion on the basis of encoder odometry information," *Sensors*, vol. 19, no. 2, p. 289, 2019.

[7] K. Forsgren, D. Shah, and D. Lum, "The road ahead for autonomous vehicles," *S&P Global-RatingsDirect. https://www. ibtta. org/sites/default/files/documents/SP% 20Global% 20Ratings*, 2018.

[8] Ruandawen and M. jian, "Path planning of restaurant delivery robot based on improved ant colony algorithm," in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, pp. 415–418, 2019.

[9] H. Habib, S. Waseem, and A. H. Ghafoor, "Development and implementation of enhanced shortest path algorithm for navigation of mobile robot for warehouse automation (mrwa)," in *2019 7th International Electrical Engineering Congress (iEECON)*, pp. 1–4, IEEE, 2019.

[10] Z. H. Khan, A. Siddique, and C. W. Lee, "Robotics utilization for healthcare digitization in global covid-19 management," *International journal of environmental research and public health*, vol. 17, no. 11, p. 3819, 2020.

[11] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, 2019.

[12] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[13] L. Kleeman *et al.*, "Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning.," in *ICRA*, pp. 2582–2587, Citeseer, 1992.

[14] S. Konduri, E. O. C. Torres, and P. R. Pagilla, "Dynamics and control of a

differential drive robot with wheel slip: application to coordination of multiple robots," *Journal of Dynamic Systems, Measurement, and Control*, vol. 139, no. 1, 2017.

[15] M. F. Jaramillo-Morales, S. Dogru, L. Marques, and J. B. Gomez-Mendoza, "Predictive power estimation for a differential drive mobile robot based on motor and robot dynamic models," in *2019 third IEEE international conference on robotic computing (IRC)*, pp. 301–307, IEEE, 2019.

[16] M. Wahab, F. Rios-Gutierrez, and A. El Shahat, *Energy modeling of differential drive robots*. IEEE, 2015.

[17] T. Lozano-Perez, "Spatial planning: A configuration space approach," in *Autonomous robot vehicles*, pp. 259–271, Springer, 1990.

[18] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." `https://www.ros.org`.

[19] G. Grisetti, C. Stachniss, and W. Burgard, "openslam_gmapping." `https://github.com/OpenSLAM-org/openslam_gmapping`, Feb. 2019.

[20] D. Lu, "ros_planning." `https://github.com/ros-planning/navigation`, Feb. 2021.

[21] A. Lampe and R. Chatila, "Performance measure for the evaluation of mobile robot autonomy," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 4057–4062, IEEE, 2006.

[22] H.-M. Huang, K. Pavek, J. Albus, and E. Messina, "Autonomy levels for unmanned systems (alfus) framework: An update," in *Unmanned Ground Vehicle*

Technology VII, vol. 5804, pp. 439–448, International Society for Optics and Photonics, 2005.

[23] D. Malyuta, "Guidance, navigation, control and mission logic for quadrotor full-cycle autonomy," Master's thesis, ETH Zurich, 2018.

[24] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4193–4198, IEEE, 2016.

[25] P. Backes, A. Diaz-Calderon, M. Robinson, M. Bajracharya, and D. Helmick, "Automated rover positioning and instrument placement," in *2005 IEEE Aerospace Conference*, pp. 60–71, IEEE, 2005.

[26] K. Yamaguchi, T. Kato, and Y. Ninomiya, "Vehicle ego-motion estimation and moving object detection using a monocular camera," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 4, pp. 610–613, IEEE, 2006.

[27] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., Iowa State University, Ames, Iowa, 1998.

[28] B. Pitzer, "usb_cam." `https://github.com/ros-drivers/usb_cam`, June 2020.

[29] J. B. Weinberg and X. Yu, "Low-cost platforms for teaching integrated systems," *Robotics & Automation Magazine*, 2003.

[30] M. Jdeed, M. Schranz, and W. Elmenreich, "A study using the low-cost swarm robotics platform spiderino in education," *Computers and Education Open*, vol. 1, p. 100017, 2021.

[31] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Choudhury, C. Mavrogiannis, and F. Sadeghi, "Mushr: A low-cost, open-source robotic racecar for education and research," *arXiv preprint arXiv:1908.08031*, 2019.

[32] Y. Yun, J. Jin, N. Kim, J. Yoon, and C. Kim, "Outdoor localization with optical navigation sensor, imu and gps," in *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 377–382, IEEE, 2012.

[33] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[34] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[35] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*, pp. 4387–4393, IEEE, 2009.

[36] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[37] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[38] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic

determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[39] S. B. Aruoba and J. Fernández-Villaverde, "A comparison of programming languages in macroeconomics," *Journal of Economic Dynamics and Control*, vol. 58, pp. 265–273, 2015.

[40] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," tech. rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[41] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter." Chapel Hill, NC, USA, 1995.

[42] N. Thacker and A. Lacey, "Tutorial: The kalman filter," *Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester*, p. 61, 1998.

[43] P. R. Belanger, P. Dobrovolny, A. Helmy, and X. Zhang, "Estimation of angular velocity and acceleration from shaft-encoder measurements," *The International Journal of Robotics Research*, vol. 17, no. 11, pp. 1225–1233, 1998.

[44] J. Biswas, "Obstacle avoidance." University Lecture, 2020.

[45] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[46] Y. Shim and G.-W. Kim, "Range sensor-based efficient obstacle avoidance through selective decision-making," *Sensors*, vol. 18, no. 4, p. 1030, 2018.

[47] S. Behnke, A. Egorova, A. Gloye, R. Rojas, and M. Simon, "Predicting away robot control latency," in *Robot Soccer World Cup*, pp. 712–719, Springer, 2003.

[48] G. Halmetschlager-Funek, M. Suchi, M. Kampel, and M. Vincze, "An empirical evaluation of ten depth cameras: Bias, precision, lateral noise, different lighting conditions and materials, and multiple sensor setups in indoor environments," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 67–77, 2018.

[49] W. Chung, S. Kim, M. Choi, J. Choi, H. Kim, C.-b. Moon, and J.-B. Song, "Safe navigation of a mobile robot considering visibility of environment," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 3941–3950, 2009.

[50] R. van der Horst and J. Hogema, "Time-to-collision and collision avoidance systems," in *6th ICTCT Workship, Salzburg*, 1994.